

IC Compiler™

Implementation

User Guide

Version J-2014.09, September 2014

SYNOPSYS®

Copyright Notice and Proprietary Information

© 2014 Synopsys, Inc. All rights reserved. This software and documentation contain confidential and proprietary information that is the property of Synopsys, Inc. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Synopsys, Inc., or as expressly provided by the license agreement.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsis and certain Synopsis product names are trademarks of Synopsis, as set forth at <http://www.synopsys.com/Company/Pages/Trademarks.aspx>.

All other product or company names may be trademarks of their respective owners.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsis does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
700 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

Copyright Notice for the Command-Line Editing Feature

© 1992, 1993 The Regents of the University of California. All rights reserved. This code is derived from software contributed to Berkeley by Christos Zoulas of Cornell University.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by the University of California, Berkeley and its contributors.
4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright Notice for the Line-Editing Library

© 1992 Simmule Turner and Rich Salz. All rights reserved.

This software is not subject to any license of the American Telephone and Telegraph Company or of the Regents of the University of California.

Permission is granted to anyone to use this software for any purpose on any computer system, and to alter it and redistribute it freely, subject to the following restrictions:

1. The authors are not responsible for the consequences of use of this software, no matter how awful, even if they arise from flaws in it.
2. The origin of this software must not be misrepresented, either by explicit claim or by omission. Since few users ever read sources, credits must appear in the documentation.
3. Altered versions must be plainly marked as such, and must not be misrepresented as being the original software. Since few users ever read sources, credits must appear in the documentation.
4. This notice may not be removed or altered.

Contents

About This Guide	xxxiv
Customer Support.....	xxxvi

Part I: IC Compiler Implementation Flow

1. Introduction to the IC Compiler Tool

IC Compiler Packages	1-2
User Interfaces	1-3
Methodology Overview	1-4
Speeding Up the Basic Flow	1-7

2. Working With the IC Compiler Tool

Starting the IC Compiler Tool	2-2
Entering <code>icc_shell</code> Commands	2-4
Getting Help on the Command Line	2-5
Checking the Syntax and Semantics of Your Scripts	2-5
Installation Requirements	2-6
Running the Synopsys Syntax Checker.....	2-7
Limitations of the Synopsys Syntax Checker.....	2-8
Bytecode-Compiled Files.....	2-9
TclPro Limitations	2-9

Working With Licenses	2-9
Listing the Licenses in Use	2-10
Getting Licenses	2-10
Enabling License Queuing	2-10
Releasing Licenses	2-11
Enabling Multicore Processing	2-12
Configuring Multithreading	2-12
Configuring Distributed Processing	2-14
Exiting the IC Compiler Tool	2-15

3. Preparing the Design

Setting Up the Libraries	3-3
Setting Up the Logic Libraries	3-3
Setting Up the Physical Libraries	3-4
Creating a Milkyway Design Library	3-5
Specifying the Milkyway Reference Libraries	3-5
Opening a Milkyway Design Library	3-7
Reporting a Milkyway Design Library	3-7
Changing Physical Library Information	3-10
Saving Physical Library Information	3-10
Verifying the Libraries	3-11
Reading the Design	3-12
Reading a Design in Milkyway Format	3-13
Reading a Design in .ddc Format	3-14
Reading a Design in ASCII Format	3-15
Handling Incomplete Data	3-16
Setting the Working Design	3-17
Setting the Current Design	3-17
Opening Designs in a Hierarchical Stack	3-17
Traversing Multiple Levels	3-18
Changing Between Different Stacks	3-19
Transforming Parent and Child Coordinates	3-19
Annotating the Floorplan Information	3-20
Reading DEF Files	3-20
Reading Floorplan Files	3-21
Copying Floorplan Information	3-21

Annotating the Scan Chain Information	3-21
Loading a SCANDEF File	3-22
Checking Scan Chain Consistency	3-23
Removing SCANDEF Data	3-23
Specifying the Power Intent	3-24
Validating the Design	3-25
Validating the Floorplan Information	3-25
Validating the Integrity of the Design Database	3-26
Creating Logical Power and Ground Connections	3-26
Creating the Connections for Single-Voltage Designs	3-27
Creating the Connections for Multivoltage Designs	3-28
Setting Up for Power Optimization	3-29
Preparing Logic Libraries for Leakage-Power Optimization	3-30
Identifying Multiple-Threshold-Voltage Cells	3-30
Analyzing the Multiple-Threshold-Voltage Cell Characteristics	3-31
Preparing Logic Libraries for Final Stage Leakage-Power Recovery	3-31
Annotating the Switching Activity	3-32
Specifying the Multiple-Threshold-Voltage Constraint	3-32
Setting Up for Multicorner-Multimode Analysis and Optimization	3-33
Defining Scenarios	3-34
Checking the Scenario Definitions	3-35
Managing the Scenarios	3-35
Selecting the Scenarios to Use	3-35
Reporting the Scenarios	3-38
Removing Scenarios	3-40
Preparing for Timing Analysis and RC Calculation	3-40
Setting Up the TLUPPlus Files	3-40
Specifying the Operating Conditions	3-42
Selecting the Operating Condition Analysis Mode	3-43
Preparing for Minimum and Maximum Timing Analysis	3-44
Setting Voltage and Temperature Scaling Between Libraries	3-46
Setting Timing Constraints	3-47
Selecting the Delay Calculation Method	3-49
Modifying the RC Coefficients for Preroute RC Estimation	3-50
Defining Net-Based Layer Constraints	3-52
Introducing Pessimism	3-52

Back-Annotating Delay or Parasitic Data	3-54
Linking Designs	3-54
Linking Designs Without Multicorner-Multimode Scenarios	3-55
Linking Designs With Multicorner-Multimode Scenarios	3-55
Resolving References With Operating Condition Mismatches	3-58
Saving a Design	3-60
Saving the Design in Milkyway Format	3-60
Saving the Design From the Command Line	3-60
Saving the Design in the GUI	3-61
Saving the Design Settings in the Milkyway Design Library	3-62
Saving the Design in ASCII Format	3-63
Writing GDSII and OASIS Layout Data Files	3-64
Closing Designs	3-65
Closing a Milkyway Design Library	3-66
Archiving Designs	3-67
4. Placement and Optimization	
Preparing for Placement and Optimization	4-2
Defining Placement Blockages	4-2
Defining Keepout Margins	4-3
Defining Area-Based Placement Blockages	4-5
Creating Placement Blockages for Thin Channels	4-10
Setting the Congestion Options	4-11
Controlling Placement Density	4-12
Defining Placement Bounds	4-13
Defining Move Bounds	4-14
Defining Group Bounds	4-15
Updating, Querying, and Removing Placement Bounds	4-16
Preventing Cell Placement in the Default Voltage Area	4-16
Defining Spacing Requirements	4-17
Defining Target Library Subsets	4-18
Defining the Buffer Strategy for Optimization	4-20
Setting the Preferred Buffers for Hold Fixing	4-21
Enabling Tie Cell Insertion	4-21
Preserving dont_touch Nets During Optimization	4-22
Limiting Fanout During Optimization	4-23

Setting Placement and Optimization Attributes	4-23
Preparing for High-Fanout Net Synthesis	4-24
Specifying Layer Optimization Settings	4-24
Creating Layer Constraints	4-25
Preparing for Clock Tree Synthesis During Placement	4-28
Inserting Port Protection Diodes	4-29
Setting Up Multithreading	4-29
Performing Placement and Optimization	4-29
Performing Placement and Optimization on Designs With Scan Chains	4-32
Performing Initial Placement on Designs With Scan Chains	4-32
Optimizing the Scan Chains	4-32
Performing Power Optimization	4-35
Setting Up for Dynamic-Power Optimization	4-35
Setting Up for Leakage-Power Optimization	4-36
Performing Low-Power Placement	4-36
Performing Leakage-Power Optimization	4-36
Performing Total-Power Optimization	4-37
Optimizing Clock-Gating Logic	4-38
Reducing the Total Negative Slack	4-38
Omitting Initial Placement	4-39
Using Physical Guidance From the Design Compiler Tool	4-39
Creating Multibit Register Banks	4-40
Preparing for Register Banking	4-41
Performing Initial Placement	4-43
Performing Register Banking	4-44
Performing Optimization	4-44
Splitting Register Banks	4-44
Outputting Additional Information During Preroute Optimization	4-44
Saving Intermediate Results During Preroute Optimization	4-45
Analyzing Placement and Optimization Results	4-46
Analyzing Placement and Optimization Feasibility	4-47
Analyzing Cell Displacement During Legalization	4-48
Reporting Cells Added During Optimization	4-50
Analyzing Cells and Nets With <code>dont_touch</code> Attributes	4-52
Reporting Unfixed Violations	4-53
Generating Physical Data Reports	4-54
Analyzing the Placement Area Utilization	4-56

Analyzing the Timing Results	4-58
Analyzing Power	4-62
Reporting Quality-of-Results	4-63
create_qor_snapshot	4-63
query_qor_snapshot.....	4-65
Analyzing Timing Results in the GUI	4-67
Applying Filters.....	4-68
Creating Subgroups	4-69
Analyzing the Clock Tree Synthesis Results.....	4-69
Analyzing the Placement in the GUI	4-70
Cell Density Map.....	4-71
Congestion Maps	4-73
Hierarchy Visual Mode	4-74
Clock Tree Visual Mode	4-75
Net Connections in Area Visual Mode	4-75
Viewing Scan Chains.....	4-79
Fine-Tuning the Placement and Optimization Results	4-79
Performing Magnet Placement	4-80
Pulling Cells in a Continuous Datapath Toward a Magnet Object.....	4-81
Excluding Cells From Magnet Placement.....	4-82
Reporting Magnet Cells	4-83
Using the Top-Level Buffering Technique for Optimization	4-83
Performing Incremental Physical Optimization	4-84
Performing Preroute Focal Optimization	4-86
Performing Layer Optimization on Existing Buffer Trees.....	4-89
Refining Placement	4-90
5. Clock Tree Synthesis	
Prerequisites for Clock Tree Synthesis	5-3
Design Prerequisites	5-3
Library Prerequisites	5-4
Handling Multivoltage Designs	5-4
Handling Hierarchical Designs	5-6
Handling Extracted Timing Models	5-6
Analyzing the Clock Trees.....	5-7

Defining the Clock Trees	5-8
Cascaded Clocks	5-8
Cascaded Generated Clocks	5-9
Non-Unate Gated Clocks	5-10
Identifying the Clock Tree Endpoints	5-11
Analyzing Clock Sink Groups	5-12
Setting Sink Group Options	5-13
Ignoring Clock Tree Exceptions	5-14
Ignoring Buffers and Inverters	5-15
Defining the Clock Root Attributes	5-16
Specifying Clock Tree Exceptions	5-17
Precedence of Clock Tree Exceptions	5-21
Specifying Nonstop Pins	5-22
Specifying Exclude Pins	5-22
Specifying Float Pins	5-23
Using Float Pins to Specify Timing for Clock Trees in Hard Macros	5-25
Specifying Float Pins for Multicorner Clock Tree Optimization	5-26
Specifying Stop Pins	5-27
Specifying Don't Touch Subtrees	5-27
Specifying Don't Buffer Nets	5-28
Specifying Don't Size Cells	5-28
Specifying Size-Only Cells	5-29
Preserving the Clock Pins of Existing Hierarchies	5-29
Handling Integrated Clock-Gating Cells	5-30
Using XOR Self-Gating to Save Dynamic Power	5-30
Setting the XOR Self-Gating Variables	5-32
Optimizing for Clock Tree QoR	5-33
Optimizing for Power	5-35
Optimizing for Timing on Enable Signals	5-37
Balancing Multiple Clocks	5-39
Defining the Delay Balancing Buffers	5-39
Defining a Clock Balance Group	5-39
Defining the Interclock Delay Requirements	5-40
Specifying the Delay Offset Requirements	5-41
Specifying the Target Insertion Delay	5-41
Using the Clock Latency Specification	5-42
Removing the Interclock Delay Settings	5-42

Specifying the Clock Tree References	5-43
Specifying Clock Tree Synthesis Options	5-45
Specifying the Clock Tree Synthesis Goals	5-49
Setting Clock Tree Design Rule Constraints	5-50
Setting the Maximum Transition Time Constraints	5-52
Setting Clock Tree Timing Goals	5-53
Setting Clock Tree Routing Options	5-54
Specifying Routing Rules	5-54
Shielding Clock Nets	5-55
Specifying Routing Layers	5-58
Specifying Routing Rules and Layers for Instance Pins and Ports	5-59
Specifying Routing Rules for Reference Cells	5-59
Inserting Boundary Cells	5-61
Selecting the Clock Tree Clustering	5-61
Enabling OCV Clustering	5-62
Controlling OCV Path Sharing	5-62
Enabling Logic-Level Balancing	5-63
Enabling Region-Aware Clock Tree Synthesis	5-66
Reducing Electromigration	5-67
Enabling Arnoldi Delay Calculation	5-69
Specifying Clock Tree Optimization Options	5-69
Controlling Embedded Clock Tree Optimization	5-70
Controlling Preroute Clock Tree Optimization	5-70
Controlling Postroute Clock Tree Optimization	5-71
Fixing DRC Violations on Clock Nets	5-71
Fixing DRC Violations on Signal Nets	5-72
Saving Intermediate Results During Preroute Optimization	5-72
Inserting User-Specified Clock Trees	5-73
Reading a Clock Configuration File	5-73
Reducing Skew Variation by Using RC Constraint-Based Clustering	5-74
Saving a Clock Configuration File	5-74
Defining the Clock Tree Structure	5-75
Handling Existing Clock Trees	5-79
Identifying Existing Clock Trees	5-79
Preserving Portions of an Existing Clock Tree	5-80
Removing Clock Trees	5-80

Verifying the Clock Trees	5-82
Implementing Clock Trees.....	5-84
Performing Multicorner-Multimode Clock Tree Synthesis and Optimization	5-91
Performing Priority-Driven Clock Tree Synthesis and Optimization	5-92
Performing Concurrent Clock and Data Optimization	5-93
Setting Up for Concurrent Clock and Data Optimization.....	5-94
Running Concurrent Clock and Data Optimization.....	5-96
Standalone Clock Tree Synthesis Capabilities.....	5-97
Performing Clock Tree Power Optimization.....	5-98
Performing Clock Tree Synthesis	5-98
High-Fanout Net Synthesis.....	5-100
Performing Clock Tree Optimization	5-101
Performing Clock Routing	5-104
Implementing Clock Meshes.....	5-105
Prerequisites for Creating Clock Meshes.....	5-106
The Clock Mesh Flow	5-107
Flattening the Logic of Integrated Clock-Gating Cells	5-108
Splitting Clock Nets.....	5-109
Creating Clock Meshes.....	5-109
Adding Clock Mesh Drivers	5-111
Routing Mesh Nets.....	5-112
Performing Quick Mesh Analysis	5-113
Compiling Premesh Trees.....	5-114
Routing Clock Nets.....	5-117
Analyzing Clock Mesh Circuits	5-118
Implementing Hierarchical Clock Meshes	5-120
Implementing Multisource Clock Trees	5-123
Multisource Clock Tree Synthesis Flow	5-124
Inserting Multisource Drivers	5-126
Building the Mesh and Premesh Tree.....	5-132
Building the Subtrees	5-132
Routing Clock Nets and Analyzing Results	5-133
H-Tree-Based Multisource Clock Trees	5-134
Analyzing the Clock Tree Results	5-134
Generating Clock Tree Reports	5-135
Generating Reports for Multicorner-Multimode Designs	5-139
Analyzing Clock Timing	5-139

Generating Categorized Timing Reports for Clock Tree Synthesis	5-140
create_qor_snapshot	5-140
query_qor_snapshot.....	5-140
Analyzing Optimization Feasibility After Clock Tree Synthesis	5-143
Analyzing Clock Trees in the GUI	5-143
Viewing Clock Trees in the Layout Window	5-145
Viewing and Setting Clock Tree Options.....	5-146
Using the Interactive Clock Tree Synthesis Window.....	5-147
Viewing the Distribution of Clock Arrival Times	5-148
Exploring Clock Tree Structures	5-148
Examining Fanout or Fanin Paths in the Clock Tree Network.....	5-151
Highlighting the Critical Paths.....	5-152
Viewing Abstract Clock Tree Graphs	5-153
Examining Clock Tree Latency	5-153
Examining Clock Tree Fanout Levels	5-154
Examining Clock Tree Overlaps	5-155
Listing Clock Tree Objects	5-156
Fine-Tuning the Clock Tree Synthesis Results	5-157
Using the Useful Skew Technique	5-157
skew_opt Flow	5-157
Balancing the Skew Using Skew Groups.....	5-159
Guidelines for Defining Skew Groups.....	5-159
Validating and Committing Skew Groups	5-161
Limitations of Using Skew Groups	5-161
Resynthesizing the Clock Trees.....	5-162
Validating the Setup	5-162
Resynthesizing the Clock Trees	5-163
Modifying the Nondefault Routing Rule	5-163
Modifying Clock Trees in the GUI.....	5-164
Inserting a Buffer	5-165
Removing a Buffer	5-165
Moving a Buffer or Gate	5-166
Reparenting a Subtree	5-166
Sizing a Buffer or Gate	5-167
Analyzing and Refining the Design.....	5-167

6. Routing Using Zroute

Zroute Features	6-3
Basic Zroute Flow	6-4
Prerequisites for Routing	6-5
Creating Design-Specific Via Masters	6-6
Creating Parameter-Based Via Masters	6-7
Creating Rectangle-Based Via Masters	6-7
Checking Routability	6-8
Setting Routing Constraints	6-11
Specifying the Routing Layers	6-13
Specifying the Global Layer Constraints	6-13
Specifying Net-Specific Routing Layer Constraints	6-15
Using Nondefault Routing Rules	6-16
Defining Nondefault Routing Rules	6-16
Applying Nondefault Routing Rules	6-24
Verifying Nondefault Routing Rules	6-25
Reporting Nondefault Routing Rules	6-25
Removing Nondefault Routing Rules	6-26
Setting the Preferred Routing Direction	6-26
Controlling Off-Grid Routing	6-27
Preventing Off-Grid Routing	6-27
Discouraging Off-Grid Routing for Vias	6-28
Defining Route Guides	6-28
Using Route Guides to Prevent Routing	6-29
Using Route Guides to Control the Routing Direction	6-30
Using Route Guides to Limit Edges in the Nonpreferred Direction	6-31
Using Route Guides to Control the Routing Density	6-31
Using Route Guides to Prioritize Routing Regions	6-31
Using Route Guides to Encourage River Routing	6-32
Using Route Guides to Fix Violations	6-33
Querying Route Guides	6-33
Removing Route Guides	6-33
Defining Routing Corridors	6-33
Reporting Routing Corridors	6-35
Modifying Routing Corridors	6-35
Removing Routing Corridors	6-36
Defining Routing Blockages	6-36

Controlling Pin Connections	6-38
Controlling Pin Tapering.....	6-40
Specifying the Tapering Method	6-40
Controlling the Tapering Width	6-41
Setting the Rerouting Mode	6-42
Setting Routing Options	6-42
Enabling Multicore Processing	6-42
Setting Zroute Options.....	6-43
Setting Common Route Options.....	6-44
Setting Global Route Options.....	6-54
Setting Track Assignment Options	6-57
Setting Detail Route Options	6-57
Setting Crosstalk Prevention Options	6-65
Setting Crosstalk Reduction Options.....	6-66
Setup-Based Net Selection.....	6-67
Hold-Based Net Selection.....	6-68
Transition-Based Net Selection.....	6-68
Static-Noise-Based Net Selection.....	6-69
Setting Multivoltage Options	6-69
Routing Clock Nets	6-72
Clock Net Routing	6-72
Clock Net Redundant Via Insertion	6-73
Clock Net Shielding	6-74
Postroute Clock Tree Optimization.....	6-74
Routing Critical Nets	6-74
Routing Secondary Power and Ground Pins	6-76
Verifying the Secondary Power and Ground Pin Attributes	6-76
Setting the Routing Constraints	6-77
Routing the Secondary Power and Ground Pins	6-78
Routing Signal Nets	6-79
Routing Signal Nets by Using Individual Routing Commands	6-80
Global Routing	6-80
Track Assignment	6-85
Detail Routing	6-86
Routing Signal Nets by Using Automatic Routing	6-91

Routing Signal Nets by Using the <code>route_opt</code> Command	6-93
Setting the Routing and Optimization Strategy.....	6-93
Using Scenario Compression for Multicorner-Multimode Designs	6-95
Running the <code>route_opt</code> Command.....	6-96
Performing Postroute Optimization	6-98
Performing Concurrent Clock and Data Optimization	6-99
Controlling the Optimization Effort.....	6-100
Fixing Hold Time Violations	6-101
Performing Leakage-Power Optimization	6-102
Performing Signal Integrity Optimization	6-103
Performing Area Recovery.....	6-103
Performing Register-to-Register Optimization.....	6-103
Performing High-Resistance Optimization.....	6-103
Controlling the Optimization Reporting	6-104
Performing Focal Optimization	6-104
Performing Final Stage Leakage-Power Recovery	6-107
Performing ECO Routing	6-110
Cleaning Up Routed Nets.....	6-112
Saving the Routing Information.....	6-113
Analyzing the Routing Results	6-113
Reporting Cell Placement and Routing Statistics	6-114
Reporting Unfixed Timing Violations	6-114
Analyzing Congestion	6-114
Generating a Congestion Report	6-114
Generating a Congestion Map	6-118
Performing Design Rule Checking Using the IC Compiler Tool.....	6-120
Performing Signoff Design Rule Checking.....	6-123
Setting Up the IC Validator Environment	6-123
Setting The Physical Signoff Options	6-124
Running the <code>signoff_drc</code> Command	6-127
Automatically Fixing Signoff DRC Violations	6-130
Verifying the Routing in an External Tool.....	6-133
Analyzing DRC Violations	6-133
Using the DRC Query Commands	6-134
Using the Error Browser	6-134

Routing Nets and Buses in the GUI	6-135
Routing Single Nets.....	6-135
Creating Physical Buses	6-137
Manual Bus Creation	6-137
Automatic Bus Creation	6-138
Modifying Buses	6-139
Routing Buses or Multiple Nets	6-140
Modifying Routed Nets	6-142
Creating Custom Wires	6-144
Postroute RC Extraction	6-146
7. Chip Finishing and Design for Manufacturing	
Overview	7-2
Inserting Tap Cells	7-3
Adding Tap Cell Arrays	7-3
Fixing Tap Spacing Violations	7-5
Removing Tap Cells.....	7-6
Finding and Fixing Antenna Violations	7-7
Defining Metal Layer Antenna Rules	7-8
Defining the Global Metal Layer Antenna Rules.....	7-8
Defining Layer-Specific Antenna Rules	7-14
Reporting Antenna Rules	7-19
Removing Antenna Rules.....	7-20
Specifying Antenna Properties	7-20
Analyzing and Fixing Antenna Violations.....	7-21
Inserting Diodes During Detail Routing	7-22
Inserting Diodes After Detail Routing	7-24
Removing Diodes	7-24
Inserting Redundant Vias	7-25
Inserting Redundant Vias on Clock Nets.....	7-25
Inserting Redundant Vias on Signal Nets	7-26
Viewing the Default Via Mapping Table	7-27
Defining a Customized Via Mapping Table.....	7-28
Postroute Redundant Via Insertion.....	7-29
Concurrent Soft-Rule-Based Redundant Via Insertion.....	7-30
Near 100 Percent Redundant Via Insertion	7-32
Preserving Timing During Redundant Via Insertion	7-33

Maximizing the Redundant Via Rate	7-33
Reporting Redundant Via Rates	7-34
Optimizing Wire Length and Via Count.	7-35
Reducing Critical Areas	7-36
Reporting Critical Areas.	7-36
Displaying Critical Area Maps	7-38
Performing Wire Spreading	7-39
Performing Wire Widening.	7-40
Shielding Nets.	7-41
Defining the Shielding Rules	7-42
Performing Preroute Shielding.	7-43
Soft Shielding Rules During Signal Routing	7-45
Performing Postroute Shielding	7-46
Shielding Example	7-46
Performing Incremental Shielding	7-47
Reporting Shielding Information	7-47
Inserting Filler Cells	7-49
Inserting Standard-Cell Fillers	7-49
Connecting Multiple Power and Ground Pins	7-51
Avoiding One-Unit-Tile Violations for Overlapping Rows	7-51
Defining the Filler Rules	7-52
Reporting Filler Cells	7-53
Removing Filler Cells	7-53
Inserting End-Cap Cells.	7-53
Inserting Well Fillers	7-55
Inserting Pad Fillers.	7-56
Inserting Metal Fill.	7-57
Setting Up the IC Validator Environment	7-58
Setting the Physical Signoff Options	7-58
Setting Up Distributed Processing.	7-59
Running the signoff_metal_fill Command	7-60
Standard Metal Fill Insertion.	7-61
Timing-Driven Metal Fill Insertion.	7-64
Metal Fill Removal	7-67
Post-ECO Metal Fill Cleanup	7-67
Analyzing the Metal Density	7-69

Part II: Advanced IC Compiler Features**8. Signal Integrity**

Analyzing and Reducing Crosstalk	8-2
Setting the Signal Integrity Options	8-3
Preventing Crosstalk During Placement	8-4
Preventing Crosstalk During Clock Tree Synthesis	8-5
Preventing and Fixing Crosstalk During Routing	8-5
Preventing Crosstalk	8-5
Fixing Crosstalk Violations During Postroute Optimization	8-6
Analyzing Crosstalk	8-7
Preparing for Crosstalk Analysis	8-7
Running Crosstalk Analysis	8-10
Example Scripts	8-12
Analyzing and Reducing Signal Electromigration	8-12
Loading Signal Electromigration Constraints	8-14
Verifying the Electromigration Constraints	8-15
Verifying the Current Unit	8-15
Validating the Design	8-15
Loading the Net Switching Information	8-15
Reading a SAIF File	8-16
Annotating the Switching Activity	8-16
Performing Signal Electromigration Analysis	8-17
Identifying Unfixable Violations	8-20
Electromigration Fixing	8-22

9. Physical Datapath With Relative Placement

Introduction to Physical Datapath With Relative Placement	9-3
Benefits of Relative Placement	9-4
Flow for Relative Placement	9-5
Methodology for the Relative Placement Flow	9-6
Script for a Relative Placement Flow	9-7
Relative Placement Flow in Design Compiler Topographical Mode	9-8
Considerations for Using Relative Placement	9-9

Creating Relative Placement Groups	9-10
Anchoring Relative Placement Groups	9-13
Specifying a Corner for an Anchored Relative Placement Group	9-15
Aligning Relative Placement Rows With the Site Row	9-17
Applying Compression to Relative Placement Groups	9-18
Handling Relative Placement Groups With Mixed Alignment	9-20
Handling Relative Placement Groups With Straddling Leaf Cells	9-20
Adding Objects to a Group	9-21
Adding Leaf Cells.	9-21
Specifying Orientations for Leaf Cells.	9-22
Aligning Leaf Cells Within a Column.	9-23
Adding a Free Placement Area for Unstructured Cells.	9-28
Orientation of Relative Placement Groups	9-30
Relative Placement Group Orientations	9-31
Relative Placement Cell Orientations	9-33
Controlling Relative Placement Cell Orientations	9-34
Adding Relative Placement Groups.	9-35
Benefits of Hierarchical Relative Placement Groups	9-36
Including Relative Placement Groups.	9-36
Instantiating Relative Placement Groups	9-38
Setting the Orientation of an Instantiated Relative Placement Group	9-40
Changing the Structures of Relative Placement Groups	9-41
Using Hierarchical Relative Placement for Straddling	9-43
Using Hierarchical Relative Placement for Compression	9-44
Effect of Ungrouping on Hierarchical Relative Placement	9-46
Effect of Uniquifying on Hierarchical Relative Placement.	9-48
Adding Keepouts	9-49
Placement and Optimization of Relative Placement Groups	9-51
Constraining the Size of a Relative Placement Group for Optimization.	9-51
Performing Relative Placement in a Design Containing Obstructions	9-51
Performing Relative Placement in a Design Containing Tap Cells.	9-52
Handling Fixed Cells During Relative Placement	9-53
Handling Physical-Only Cells During Relative Placement	9-54
Using Move Bounds to Constrain Relative Placement	9-55
Supporting Relative Placement Groups in Virtual Flat Placement	9-55
Propagating Relative Placement Groups in Physical Hierarchy	9-57

Postplacement for Relative Placement Groups	9-58
Adding Incremental Relative Placement Groups.....	9-58
Controlling Movement When Legalizing Relative Placement Groups	9-59
Ignoring Site Rows When Legalizing Relative Placement Groups	9-59
Legalizing Relative Placement Groups in a Placed Design.....	9-60
Exposing Unused Space in Relative Placement Groups.....	9-62
Postplacement Optimization of Relative Placement Groups.....	9-62
Preserving Relative Placement Structures	9-62
Optimizing Relative Placement Cells.....	9-64
Buffering Strategy for Relative Placement Groups	9-64
Analyzing the Relative Placement Results	9-65
Checking Relative Placement Constraints.....	9-65
Checking Relative Placement Groups Before Placement.....	9-67
Locating a Relative Placement Group	9-68
Analyzing Relative Placement Groups in the GUI	9-68
Working With Relative Placement Groups in the GUI.....	9-69
Using the Relative Placement Hierarchy Browser.....	9-69
Viewing Relative Placement Groups	9-70
Viewing Relative Placement Group Net Connections	9-71
Viewing Net Connectivity Between Relative Placement Groups	9-72
Moving Relative Placement Groups.....	9-73
Editing and Creating Relative Placement Groups	9-73
Using the Relative Placement Window	9-75
Viewing the Hierarchy of Relative Placement Groups	9-76
Viewing Logical Connections in a Relative Placement Group	9-77
Editing Relative Placement Groups in a Relative Placement View Window	9-80
Editing the Structures of Relative Placement Groups	9-81
Querying Relative Placement Groups and Objects	9-83
Displaying Relative Placement Group Attributes.....	9-84
Displaying Relative Placement Cell Attributes	9-85
Displaying Relative Placement Keepout Attributes	9-86
Reporting the Total Net Lengths of Relative Placement Groups	9-86
Querying Relative Placement Groups	9-87
Querying Objects in Relative Placement Groups	9-87
Query Commands for Relative Placement Groups and Objects	9-89

Saving Relative Placement Information	9-90
Ignoring Relative Placement Constraints	9-91
Removing Relative Placement Groups	9-92
Changing Relative Placement Information	9-93
Reporting the Attributes of a Relative Placement Group	9-93
Changing the Options of a Relative Placement Group	9-93
Setting Relative Placement Group Attributes	9-94
Removing Relative Placement Group Attributes	9-97
Renaming a Group	9-98
Removing Objects From a Group	9-99
Changing Specific Objects Within a Group	9-99
Changing the Attributes of Relative Placement Cells	9-100
Changing the Dimensions of Relative Placement Keepouts	9-100
Deriving Relative Placement Groups	9-100
Extracting a Relative Placement Group	9-101
Ordering Extracted Relative Placement Groups	9-102
Summary of Relative Placement Commands	9-103
Limitations of Relative Placement	9-104

10. Using Hierarchical Models

Overview of Block Abstractions	10-2
Supported Flows With Block Abstractions	10-3
Creating Block Abstractions	10-4
Creating a Block Abstraction for Multicorner-Multimode Usage	10-5
Using Block Abstractions	10-5
Linking to Block Abstractions	10-6
Reporting and Querying Block Abstractions	10-6
Checking the Readiness of Block Abstractions	10-7
Applying Top-Level Constraints	10-8
Controlling the Extent of Logic Loaded for Block Abstractions	10-8
Ignoring Timing Paths Entirely Within Block Abstractions	10-9
Using Block Abstractions in a Multicorner-Multimode Flow	10-10
Handling Nested Physical Hierarchy	10-11
Viewing Block Abstractions in the GUI	10-12

Performing Top-Level Optimization	10-12
Performing Transparent Interface Optimization	10-13
Setting Up for Transparent Interface Optimization	10-15
Specifying Block-Specific Size-Only Settings	10-16
Customizing the Commands Executed During Block Update	10-17
Preserving Relative Placement Groups Inside Hierarchical Blocks	10-18
Preserving the Route Properties of Nets Inside Hierarchical Blocks	10-18
Skipping Automatic Block Updates	10-18
Generating ECO Files	10-19
Setting Up the Distributed Processing Environment for Block Updates	10-19
Reporting the Transparent Interface Optimization Settings	10-20
Example Scripts	10-20

11. In-Design Rail Analysis

Overview	11-3
Prerequisites for In-Design Rail Analysis	11-6
License Requirements	11-6
Library Requirements	11-7
Preparing for In-Design Rail Analysis	11-9
Setting the Rail Analysis Options	11-9
Specifying the PrimeRail Executable File	11-10
Specifying the Working Directory	11-10
Setting the Analysis Mode	11-10
Defining the Ideal Voltage Sources	11-11
Setting the Operating Voltage	11-12
Controlling Pin Shape Derivation During PG Extraction	11-13
Summary of the Rail Analysis Options	11-13
Selecting the Scenario	11-17
Enabling Multicore Processing	11-18
Performing Rail Integrity Checking	11-18
Defining the Rail Integrity Layout Check Strategies	11-20
Defining a Strategy for the Floating Shapes Check	11-20
Defining a Strategy for the Floating Pin Shapes Check	11-21
Defining a Strategy for the Dangling Pin Access Edge Check	11-22
Defining a Strategy for the Dangling Vias Check	11-22
Defining a Strategy for the Missing Vias Check	11-23
Defining a Strategy for the Discontinuous Connections Check	11-26

Controlling the Tracing Behavior	11-27
Filtering the Errors Based on Area	11-28
Running the Rail Integrity Checking	11-28
Rail Integrity Checking Example	11-29
Performing Rail Integrity Checking in the GUI	11-30
Integration With Template-Based Power Network Synthesis	11-31
Performing Voltage Drop Analysis	11-32
Defining the Switching Activity Information	11-33
Defining the Analysis Scope	11-34
Enabling Effective Voltage Drop Reporting	11-34
Reusing Rail Analysis Data	11-35
Running the Voltage Drop Analysis	11-36
Performing Electromigration Analysis	11-39
Defining the Electromigration Rules	11-39
Running the Electromigration Analysis	11-41
Performing Inrush Current Analysis	11-43
Generating a PrimeRail Configuration File	11-44
Running the Inrush Current Analysis	11-45
Performing Minimum Path Resistance Analysis	11-47
Running the Minimum Path Resistance Analysis	11-47

12. Multivoltage Design Flow

Multivoltage Designs	12-3
Power Domains	12-3
Voltage Areas	12-4
Associating Power Domains and Voltage Areas	12-4
Level-Shifter and Isolation Cells	12-5
Physical Implementation	12-5
High-Level Design Flow for Multivoltage Designs	12-6
UPF Flow for Multivoltage Designs	12-8
Golden UPF Flow	12-9
Implementing Power Switches	12-11
Connecting Power and Ground Nets	12-12
Explicitly Identifying Power and Ground Nets	12-13

Specifying the Operating Voltage	12-13
Defining and Using Voltage Areas	12-14
Creating Voltage Areas	12-16
Updating Voltage Areas in a Design	12-17
Default Voltage Area	12-17
Including Guard Bands With Voltage Areas.....	12-18
Handling Nested Voltage Areas.....	12-19
Voltage Area Support for Macros.....	12-20
Defining Always-On Power Wells Within Voltage Areas	12-21
Removing Voltage Areas From the Design	12-21
Reporting on the Voltage Areas of a Design	12-21
Power Management Cells and UPF Strategies.....	12-22
Managing Level-Shifter Cells	12-22
Inserting Buffer-Type Level Shifters	12-23
Setting the Threshold Voltage for Level Shifters	12-25
Defining the Level-Shifter Strategy	12-26
Associating Specific Library Cells With a Level-Shifter Strategy	12-27
Checking Level-Shifter Violations	12-27
Placing Level Shifters	12-27
Removing Level Shifters.....	12-27
Managing Isolation and Enable-Type Level-Shifter Cells.....	12-28
Inserting Isolation Cells and Enable-Type Level Shifters	12-28
Defining the Isolation Strategy	12-28
Retention Registers	12-29
Always-On Logic	12-30
Power Management Cells and the ASCII UPF Flow	12-30
Inserting Power Management Cells	12-30
Associating of Power Management Cells With UPF Strategies	12-31
N-Well and P-Well Bias.....	12-31
Example IC Compiler Script for the UPF Flow	12-32
Voltage-Area-Aware Capabilities	12-34
Automatic High-Fanout Synthesis	12-34
Virtual Hierarchy Routing.....	12-35
Voltage-Area-Based Optimization	12-35
Voltage-Area-Based Clock Tree Synthesis and Optimization	12-36
Voltage Area Feedthrough Flow	12-36

Voltage-Area-Aware Always-On Synthesis	12-37
Controlling the Dummy Hierarchy	12-38
Voltage-Area-Based Power and Ground Net Associations	12-39
Hierarchical Flow	12-39
Multivoltage Relative Placement Capability	12-40
Placing and Optimizing the Design	12-40
Clock Tree Synthesis in Power-Down Domains	12-41
Multivoltage-Specific Queries, Checks, and Reports	12-42
Querying and Reporting Multivoltage-Specific dont_touch Attributes	12-43
Multivoltage-Specific Checks	12-43
Multivoltage-Specific Reports	12-44
Hierarchical UPF Flow	12-46
Design Compiler to IC Compiler Interface	12-47
Guidelines for the Hierarchical UPF Flow	12-47
Methodology Guidelines	12-48
Interface Guidelines	12-48
Implementation Guidelines	12-48
Limitations	12-50
Full-Chip Design Planning	12-50
Block-Level Place and Route	12-53
Top-Level Place and Route	12-54
13. ECO Flow	
Unconstrained ECO Flow	13-3
Freeze Silicon ECO Flow	13-3
Freeze Silicon QoR Tuning Flow	13-4
Hierarchical Placement Area Reservation ECO Flow	13-7
Top-Down Hierarchical Placement Area Reservation ECO Flow	13-7
Bottom-Up Hierarchical Placement Area Reservation ECO Flow	13-8
Adding Spare Cells	13-9
Adding Spare Cells by Instantiating in the Verilog Netlist	13-9
Automatically Placing Spare Cells During Placement	13-10
Manually Placing Spare Cells Before Placement	13-10
Manually Placing Spare Cells After Placement	13-11

Adding Spare Cells by Using the <code>insert_spare_cells</code> Command	13-12
Adding Programmable Spare Cells	13-13
Making ECO Changes	13-13
Importing a Verilog Netlist	13-13
Using Netlist Editing Tcl Commands	13-14
Reverting Sized Cells	13-16
Placing Cells During the Unconstrained ECO Flow	13-16
Placing and Mapping Cells During the Freeze Silicon ECO Flow	13-17
Adding Repeater Cells on Routed Nets	13-18
Performing ECO Routing	13-20
Limiting ECO Routing to Specific Layers	13-21
Reporting ECO History and Physical Changes	13-22
Comparing Designs	13-22

14. In-Design Signoff Timing Analysis and Correlation Checking

In-Design Signoff Timing Analysis	14-2
Key Features	14-2
Running In-Design Signoff Timing Analysis	14-3
Correlating Multicorner-Multimode Scenarios	14-7
Mapping Scenarios Between the IC Compiler and PrimeTime Tools	14-8
Enabling On-Chip Variation	14-9
Enabling Static Noise Analysis	14-9
Setting Up the PrimeTime Tool	14-10
Modifying the PrimeTime Settings	14-10
Enabling Consistency Settings	14-13
Configuring PrimeTime Distributed Processing	14-13
Setting Up the StarRC Tool	14-15
Modifying the StarRC Settings	14-15
Configuring StarRC Distributed Processing	14-18
Automatically Generating the PrimeTime and StarRC Settings	14-18
Running the <code>run_signoff</code> Command	14-20
Using A PrimeTime Saved Session To Skip Initial Extraction	14-22
Previewing Slack Improvements	14-23
Troubleshooting and Problem Solving	14-25
Checking the Signoff Correlation	14-25

Appendix A. Using GUI Tools

Working With the GUI	A-2
Opening the GUI	A-2
Using GUI Windows	A-4
Menu Bar	A-5
Toolbars	A-5
Status Bar	A-6
View Windows	A-6
Panels	A-7
Working in the Main Window	A-8
Choosing Menu Commands in GUI Windows	A-9
Finding Menu Commands and Dialog Boxes	A-10
Previewing <code>icc_shell</code> Commands in Dialog Boxes	A-12
Displaying the List of Keyboard Shortcuts	A-12
Working With the Console	A-12
Viewing the Session Log	A-13
Viewing the History View	A-15
Using Online Help	A-17
Using the Man Page Viewer	A-17
Using Contextual Help	A-18
Using the Help System	A-18
Viewing the Physical Layout	A-19
Setting the Current Task	A-21
Setting the Primary Design	A-22
Setting GUI Preferences	A-22
Recording and Replaying a GUI Session	A-23
Closing the GUI	A-23
Exiting the IC Compiler Tool From the GUI	A-24
Viewing a Design Interactively	A-24
Selecting a Mouse Tool	A-26
Previewing Objects	A-28
Selecting Objects	A-29
Viewing Object Information	A-31
Highlighting Objects	A-33
Selecting Only the Highlighted Objects	A-36
Selecting or Highlighting Objects by Name	A-36

Magnifying and Traversing the Design.....	A-37
Following Selected Objects	A-40
Reusing Zoom and Pan Settings	A-40
Redrawing the Active View	A-41
Examining Design and Object Information	A-41
Viewing and Editing Object Properties	A-42
Exploring the Design Hierarchy	A-44
Examining Hierarchical Cells	A-45
Examining Hierarchical Cell Placement	A-45
Exploring the Virtual Hierarchy of a Flat Design	A-46
Examining Virtual Hierarchical Cells.....	A-47
Examining Virtual Hierarchical Cell Placement	A-49
Managing Attribute Groups	A-50
Searching for Objects by Name.....	A-51
Selecting and Viewing Timing Paths	A-53
Viewing Reports.....	A-54
Saving an Image of a Window or View	A-55
Viewing the Physical Design.....	A-56
Navigating Through Layout Views	A-57
Displaying Grid Lines.....	A-58
Drawing Rulers	A-58
Adjusting the Color Brightness	A-60
Displaying Cell Orientations.....	A-60
Setting and Saving View Properties	A-61
Saving and Restoring Layout View Settings.....	A-65
Applying Predefined Layout View Settings	A-65
Viewing Design Overlays.....	A-66
Using Stroke Activated Commands	A-68
Analyzing the Physical Design	A-69
Using Visual Analysis Modes.....	A-70
Displaying Flylines	A-71
Displaying Net Connectivity	A-72
Using Visual Modes	A-72
Using Map Modes.....	A-75
Tracing Paths Through the Design.....	A-76
Analyzing the Data Flow for Macro Placement	A-79
Examining and Editing Relative Placement Groups	A-81

Analyzing Signal Integrity	A-82
Analyzing Clock Trees	A-83
Analyzing Timing Paths	A-85
Opening a Timing Analysis Window	A-85
Viewing Timing Path Details	A-86
Using Timing Analysis Views.	A-87
Viewing Histograms	A-87
Examining Path Profiles	A-89
Viewing Path and Design Schematics	A-89
Inspecting a Timing Path	A-92
Examining Routing and Verification Errors	A-94
Viewing and Fixing Errors	A-96
Filtering Errors in the Error Browser	A-98
Selecting Errors in the Layout View	A-100
Examining Nets Associated With Detail Routing Errors	A-101
Examining Net Nodes Associated With Open Locator Errors	A-101
Examining Errors in an Area	A-102
Saving the Error List in an Error Report File	A-102
Saving Error Updates in an Error View	A-103
Saving a Copy of the Editor DRC Error View.	A-104
Editing the Physical Layout.	A-104
Editing Objects Interactively.	A-107
Moving, Resizing, Modifying, and Removing Objects.	A-109
Expanding and Rotating Objects	A-111
Routing Nets and Physical Buses.	A-112
Creating Floorplan Objects.	A-114
Working in the Editing Environment.	A-115
Reversing and Reapplying Edit Changes	A-116
Fixing and Unfixing Objects for Edits	A-118
Changing the Way Objects Snap to a Grid.	A-118
Setting Editing Options	A-120
Editing Groups of Objects.	A-124
Viewing Estimated Resistance Values for Wires, Vias, and Pins	A-124
Finding and Fixing Editor DRC Violations	A-126
Verifying Route Edits Interactively	A-129
Verifying Edited Nets	A-129

Setting Editor DRC Options	A-131
Examining Editor DRC Errors	A-132
Editing Hierarchical Cells in Place	A-133
Annotating the Layout	A-137

Index

Preface

This preface includes the following sections:

- [About This Guide](#)
- [Customer Support](#)

About This Guide

The Synopsys IC Compiler™ tool provides a complete netlist-to-GDSII or netlist-to-clock-tree-synthesis design solution, which combines proprietary design planning, physical synthesis, clock tree synthesis, and routing for logical and physical design implementations throughout the design flow.

This guide describes the IC Compiler implementation and integration flow; the companion volume, *IC Compiler Design Planning User Guide*, describes the design planning flow.

Audience

This user guide is for design engineers who use IC Compiler to implement designs.

To use IC Compiler, you need to be skilled in physical design and design synthesis and be familiar with the following:

- Physical design principles
 - The Linux or UNIX operating system
 - The Tool Command Language (Tcl)
-

Related Publications

For additional information about the IC Compiler tool, see the documentation on the Synopsys SolvNet® online support site at the following address:

<https://solvnet.synopsys.com/DocsOnWeb>

You might also want to see the documentation for the following related Synopsys products:

- Design Compiler®
- Milkyway Environment™
- IC Validator
- Hercules™
- PrimeRail
- PrimeTime® Suite

Release Notes

Information about new features, changes, enhancements, known limitations, and resolved Synopsys Technical Action Requests (STARs) is available in the *IC Compiler Release Notes* on the SolvNet site.

To see the *IC Compiler Release Notes*,

1. Go to the SolvNet Download Center located at the following address:

<https://solvnet.synopsys.com/DownloadCenter>

2. Select IC Compiler, and then select a release in the list that appears.
-

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
Courier	Indicates syntax, such as <code>write_file</code> .
<i>Courier italic</i>	Indicates a user-defined value in syntax, such as <code>write_file design_list</code> .
Courier bold	Indicates user input—text you type verbatim—in examples, such as <code>prompt> write_file top</code>
[]	Denotes optional arguments in syntax, such as <code>write_file [-format fmt]</code>
...	Indicates that arguments can be repeated as many times as needed, such as <code>pin1 pin2 ... pinN</code> .
	Indicates a choice among alternatives, such as <code>low medium high</code>
Ctrl+C	Indicates a keyboard combination, such as holding down the Ctrl key and pressing C.
\	Indicates a continuation of a command line.
/	Indicates levels of directory structure.
Edit > Copy	Indicates a path to a menu command, such as opening the Edit menu and choosing Copy.

Customer Support

Customer support is available through SolvNet online customer support and through contacting the Synopsys Technical Support Center.

Accessing SolvNet

The SolvNet site includes a knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. The SolvNet site also gives you access to a wide range of Synopsys online services including software downloads, documentation, and technical support.

To access the SolvNet site, go to the following address:

<https://solvnet.synopsys.com>

If prompted, enter your user name and password. If you do not have a Synopsys user name and password, follow the instructions to sign up for an account.

If you need help using the SolvNet site, click HELP in the top-right menu bar.

Contacting the Synopsys Technical Support Center

If you have problems, questions, or suggestions, you can contact the Synopsys Technical Support Center in the following ways:

- Open a support case to your local support center online by signing in to the SolvNet site at <https://solvnet.synopsys.com>, clicking Support, and then clicking “Open A Support Case.”
- Send an e-mail message to your local support center.
 - E-mail support_center@synopsys.com from within North America.
 - Find other local support center e-mail addresses at
<http://www.synopsys.com/Support/GlobalSupportCenters/Pages>
- Telephone your local support center.
 - Call (800) 245-8005 from within North America.
 - Find other local support center telephone numbers at
<http://www.synopsys.com/Support/GlobalSupportCenters/Pages>

Part I: IC Compiler Implementation Flow

1

Introduction to the IC Compiler Tool

The IC Compiler tool is a single, convergent netlist-to-GDSII or netlist-to-clock-tree-synthesis design tool for chip designers developing very deep submicron designs. It takes as input a gate-level netlist, a detailed floorplan, timing constraints, physical and timing libraries, and foundry-process data, and it generates as output either a GDSII-format file of the layout or a Design Exchange Format (DEF) file of placed netlist data ready for a third-party router. The IC Compiler tool can also output the design at any time as a binary Synopsys Milkyway database for use with other Synopsys tools based on Milkyway or as ASCII files (Verilog, DEF, and timing constraints) for use with tools not from Synopsys.

This chapter contains the following sections:

- [IC Compiler Packages](#)
- [User Interfaces](#)
- [Methodology Overview](#)

IC Compiler Packages

The IC Compiler tool has five packages: IC Compiler, IC Compiler-AG, IC Compiler-XP, IC Compiler-PC, and IC Compiler-DP.

- If you are using the IC Compiler tool for a netlist-to-clock-tree-synthesis design flow and will perform routing with another tool, you need the IC Compiler-PC package.
- If you are also performing routing (netlist-to-GDSII flow), you need the IC Compiler package.
- If you are designing at process nodes of 130 nm and above and need a complete netlist-to-GDSII flow, you might consider the IC Compiler-XP package.
- If you are using the IC Compiler tool only to perform design planning, use the IC Compiler-DP package.
- If you are using the IC Compiler tool for both design planning and implementation, including routing, for process nodes below 130 nm, use the IC Compiler package, which contains all of the IC Compiler functionality.
- If you are using the IC Compiler tool for process nodes at or below 20 nm, use the IC Compiler-AG package, which contains all of the IC Compiler functionality plus support for double patterning and advanced node design rules.

Note:

The design planning functionality is described in the *IC Compiler Design Planning User Guide*.

The IC Compiler, IC Compiler-AG, IC Compiler-PC, and IC Compiler-DP packages support both Zroute and the classic router. For these packages, Zroute is the default router. The IC Compiler-XP package supports only the classic router. This document assumes the use of Zroute for routing and chip finishing. For information about using the classic router, see the *IC Compiler Classic Router User Guide*.

Basic power optimization capabilities, including multithreshold leakage optimization, are available in the IC Compiler, IC Compiler-AG, and IC Compiler-XP packages. Advanced power optimization capabilities, including power-aware placement and low-power clock tree synthesis, are available only in the IC Compiler and IC Compiler-AG packages. You can enable both the basic and advanced power optimization capabilities in the IC Compiler-PC package by adding the optional Power Compiler license.

Multivoltage and multithreshold-CMOS (MTCMOS) related low-power design capabilities are available only in the IC Compiler, IC Compiler-AG, IC Compiler-PC, and IC Compiler-DP packages. These capabilities are standard in the IC Compiler-PC package and do not require a Power Compiler license.

In addition, concurrent multicorner-multimode (MCMM) timing analysis and relative placement are available only in the IC Compiler, IC Compiler-AG, IC Compiler-PC, and IC Compiler-DP packages.

Advanced clock tree synthesis capabilities, such as multicorner-multimode clock tree synthesis, clock mesh synthesis, and clock-sink clustering based on on-chip variation (OCV) analysis are available only in the IC Compiler, IC Compiler-AG, and IC Compiler-PC packages.

The following advanced features are available only in the IC Compiler and IC Compiler-AG packages: In-Design static timing analysis and physical verification, advanced design rule support for process nodes below 130 nm, and advanced design-for-yield (DFY) capabilities.

User Interfaces

The IC Compiler tool provides two user interfaces:

- Shell interface (icc_shell) – The IC Compiler command-line interface is used for scripts, batch mode, and push-button operations.

Basic operations for the command-line interface are described in [Chapter 2, “Working With the IC Compiler Tool.”](#)

- Graphical user interface (GUI) – The IC Compiler GUI is an advanced analysis and physical editing tool. Certain tasks, such as very accurately displaying the design and providing visual analysis tools, can be performed only from the GUI. The look and feel of the IC Compiler GUI is consistent with the look and feel of other Synopsys GUIs.

The GUI and how to use it are described in [Appendix A, “Using GUI Tools.”](#)

The IC Compiler tool uses the Tool Command Language (Tcl), which is used in many applications in the EDA industry. Using Tcl, you can extend the IC Compiler command language by writing reusable procedures and scripts. For general information about Tcl and the command-line interface, see the *Using Tcl With Synopsys Tools* manual.

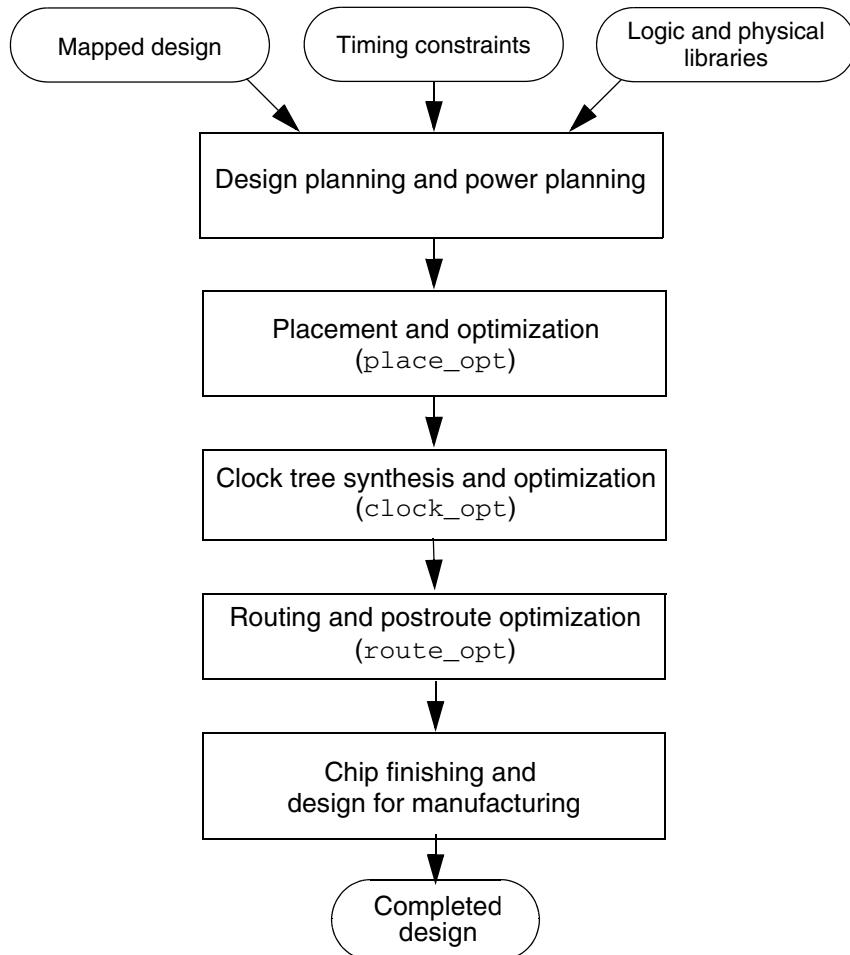
Methodology Overview

The IC Compiler design flow is an easy-to-use, single-pass flow that provides convergent timing closure. [Figure 1-1](#) shows the basic IC Compiler design flow, which is centered around three core commands that perform placement and optimization (`place_opt`), clock tree synthesis and optimization (`clock_opt`), and routing and postroute optimization (`route_opt`).

Note:

If you have the IC Compiler-PC package, you cannot perform routing and postroute optimization or chip finishing and design for manufacturing tasks in the IC Compiler tool.

Figure 1-1 IC Compiler Design Flow



For most designs, the `place_opt`, `clock_opt`, and `route_opt` steps are preset for optimal results. The IC Compiler tool also provides additional placement, clock tree synthesis, and routing technologies, as well as extended physical synthesis tools, that you can use to further improve the quality of results for your design.

To run the IC Compiler design flow,

1. Set up the libraries and prepare the design data, as described in [Chapter 3, “Preparing the Design.”](#)
2. Perform design planning and power planning.

When you perform design planning and power planning, you create a floorplan to determine the size of the design, create the boundary and core area, create site rows for the placement of standard cells, set up the I/O pads, and create a power plan.

For more information about design planning and power planning, see the *IC Compiler Design Planning User Guide*.

3. Perform placement and optimization.

To perform placement and optimization, use the `place_opt` core command (or choose Placement > Core Placement and Optimization in the GUI).

The `place_opt` core command addresses and resolves timing closure for your design. This iterative process uses enhanced placement and synthesis technologies to generate legalized placement for leaf cells and an optimized design. You can supplement this functionality by optimizing for power, recovering area for placement, minimizing congestion, and minimizing timing and design rule violations.

For more information about placement and optimization, see [Chapter 4, “Placement and Optimization.”](#)

4. Perform clock tree synthesis and optimization.

To perform clock tree synthesis and optimization, use the `clock_opt` core command (or choose Clock > Core CTS and Optimization in the GUI).

IC Compiler clock tree synthesis and embedded optimization solve complicated clock tree synthesis problems, such as blockage avoidance and the correlation between preroute and postroute data. Clock tree optimization improves both clock skew and clock insertion delay by performing buffer sizing, buffer relocation, gate sizing, gate relocation, level adjustment, reconfiguration, delay insertion, dummy load insertion, and balancing of interclock delays.

For more information about clock tree synthesis and optimization, see [Chapter 5, “Clock Tree Synthesis.”](#)

5. Perform routing and postroute optimization.

To perform routing and postroute optimization, use the `route_opt` core command (or choose Route > Core Routing and Optimization in the GUI).

As part of routing and postroute optimization, the IC Compiler tool performs global routing, track assignment, detail routing, topological optimization, and engineering change order (ECO) routing. For most designs, the default routing and postroute optimization setup produces optimal results. If necessary, you can supplement this functionality by optimizing routing patterns and reducing crosstalk or by customizing the routing and postroute optimization functions for special needs.

For more information about routing and postroute optimization, see [Chapter 6, “Routing Using Zroute.”](#)

6. Perform chip finishing and design for manufacturing tasks, as described in [Chapter 7, “Chip Finishing and Design for Manufacturing.”](#)

The IC Compiler tool provides chip finishing and design for manufacturing and design for yield capabilities that you can apply throughout the various stages of the design flow to address process design issues encountered during chip manufacturing.

7. Save the design.

Save your design in the Milkyway format. This format is the internal database format used by the IC Compiler tool to store all the logical and physical information about a design. For more information, see [“Saving the Design in Milkyway Format” on page 3-60.](#)

The IC Compiler reference methodology (<https://solvnet.synopsys.com/rmgen>) provides scripts that implement this flow.

Speeding Up the Basic Flow

If runtime is a concern and a slight degradation in quality of results is acceptable, you can run a low-effort quick flow that uses a low effort for placement and optimization and a low effort for routing and postroute optimization. This quick flow can save 30 to 50 percent in runtime, but it can also slightly degrade the quality of results.

To run a quick flow, run the `set_fast_mode true` command before running the core commands. When you enable fast mode,

- The `place_opt` command uses low effort (and ignores the `-effort` option) and invokes congestion removal.

To further reduce runtime, you can disable congestion removal by specifying the `-no_congestion` option.

Note:

The quality of results after running the `place_opt` command with low effort can be disappointing, but it should improve after running clock tree synthesis.

- There is no impact on the `clock_opt` command.
- The `route_opt` command uses low effort (and ignores the `-effort` option).

2

Working With the IC Compiler Tool

This chapter describes standard tasks for working with the IC Compiler tool.

- Starting the IC Compiler Tool
- Entering `icc_shell` Commands
- Getting Help on the Command Line
- Checking the Syntax and Semantics of Your Scripts
- Working With Licenses
- Enabling Multicore Processing
- Exiting the IC Compiler Tool

Starting the IC Compiler Tool

The IC Compiler tool operates in the X Windows System on UNIX or Linux. Before starting the tool, make sure the path to the bin directory is included in your \$PATH variable.

You start IC Compiler by entering the `icc_shell` command in a UNIX or Linux shell:

```
% icc_shell
```

By default, this command starts the tool in the command-line interface (`icc_shell`) using the IC Compiler package.

- To start the tool in the GUI instead, use the `-gui` option. For information about using the GUI, see [Appendix A, “Using GUI Tools.”](#)

```
% icc_shell -gui
```

- To start the tool using a package other than the IC Compiler package, use the option shown in [Table 2-1](#). For information about the IC Compiler packages, see [“IC Compiler Packages” on page 1-2](#).

Table 2-1 Command Options for Starting the IC Compiler Packages

IC Compiler package	icc_shell command option
IC Compiler-AG	<code>-ag_mode</code>
IC Compiler-XP	<code>-xp_mode</code>
IC Compiler-PC	<code>-psyn_mode</code>
IC Compiler-DP	<code>-dp_mode</code>

You can include other options on the command line when you start the tool. For example, you can use

- `-f script_file_name` to execute a script
- `-x command` to execute an IC Compiler shell command
- `-display terminal_name` to display the GUI on a different terminal from the one you set with the DISPLAY environment variable
- `-h` to display a list of the available options (without starting the tool)

For a complete list of startup options, see the man page.

At startup, the tool performs the following tasks:

1. Creates a command log file

The command log file records all `icc_shell` commands processed by the tool, including those in the setup files. By default, the tool writes the command log to a file named `command.log` in the directory from which you invoked `icc_shell`. You can change the name of the command log file by setting the `sh_command_log_file` variable in your setup file.

Note:

You must make any changes to this variable before you start the tool. If your setup file does not contain this variable, the tool automatically creates the `command.log` file.

The tool also creates a file named `icc_output.txt` that contains a log of the text shown in `icc_shell` during the session. This text file is not executable.

For more information about the command log file, see *Using Tcl With Synopsys Tools*.

2. Runs the setup files

The IC Compiler tool runs both the synthesis setup files, named `.synopsys_dc.setup`, and the `icc_shell` setup files, named `.synopsys_icc.tcl`. You can have up to three of each of these setup files: one in the installation (Synopsys root) directory, one in your home directory, and one in the project directory (the current working directory in which you start the tool).

If settings conflict between setup files with the same name, the tool gives precedence to the settings in your project directory first, then in your home directory, and finally in the installation directory. This order of precedence allows you to define your general personal settings in your home directory and the design-related settings in each project directory.

For more information about setup files, see *Using Tcl With Synopsys Tools*.

3. Runs any script files or commands specified by the `-f` and `-x` options, respectively, on the command line

4. Displays the program header and `icc_shell>` prompt in the shell in which you started `icc_shell`

The program header lists all IC Compiler packages for which your site is licensed.

Entering `icc_shell` Commands

You interact with the tool by using `icc_shell` commands, which are based on the Tool Command Language (Tcl) and include certain command extensions needed to implement specific IC Compiler functionality. The IC Compiler command language provides capabilities similar to UNIX command shells, including variables, conditional execution of commands, and control flow commands. You can

- Enter individual commands interactively at the `icc_shell>` prompt in `icc_shell`
- Enter individual commands interactively on the console command line in the GUI
- Run one or more Tcl command scripts, which are text files containing `icc_shell` commands

When entering a command, an option, or a file name, you can minimize your typing by pressing the Tab key when you have typed enough characters to specify a unique name; IC Compiler completes the remaining characters. If the characters you typed could be used for more than one name, IC Compiler lists the qualifying names, from which you can select by using the arrow keys and the Enter key.

For general information about using the command-line interface and using Tcl with the IC Compiler tool, see *Using Tcl With Synopsys Tools*.

Getting Help on the Command Line

The following online information resources are available while you are using `icc_shell`:

- Brief command help

To display a brief description of an `icc_shell` command,

- Enter `help` followed by the command name:

```
icc_shell> help command_name
```

- Command help, which is a list of options and arguments used with a specified `icc_shell` command

To display a list of command options and arguments,

- Enter the command name followed by the `-help` option:

```
icc_shell> command_name -help
```

- Man pages

To display a man page in `icc_shell`,

- Enter `man` followed by the command or variable name in `icc_shell`:

```
icc_shell> man command_or_variable_name
```

Checking the Syntax and Semantics of Your Scripts

The Synopsys Syntax Checker, which is based on the TclPro Checker (`procheck`), helps you find syntax and semantic errors in your Tcl scripts. Everything that you need for syntax and semantic checking is shipped with the IC Compiler tool. In this environment, the following items are checked:

- Unknown options
- Ambiguous option abbreviation
- Use of exclusive options
- Missing required options
- Validation of literal option values for numerical range (range, `<=`, `>=`)
- Validation of one-of-string (keyword) options

- Recursion into constructs that have script arguments, such as the `redirect` and `foreach_in_collection` commands
- Warning of duplicate options that override previous values

The open-source TclPro toolkit also includes the following tools, which are not included in the IC Compiler distribution, but which can be downloaded separately:

- TclPro Compiler (`procomp`), which is a standalone bytecode compiler
- TclPro Debugger (`prodebug`), which is a Tcl debugger

For more information about the TclPro tools, see the following Web address:

<http://tcl.sourceforge.net>

The following sections discuss how to use these tools in the Synopsys environment:

- [Installation Requirements](#)
- [Running the Synopsys Syntax Checker](#)
- [Limitations of the Synopsys Syntax Checker](#)
- [Bytecode-Compiled Files](#)
- [TclPro Limitations](#)

Installation Requirements

To access only the syntax checker, you do not need to download the TclPro tools. To use any TclPro tools other than the syntax checker, you must install TclPro on your system. After you have installed TclPro on your system, you must define the `SNPS_TCLPRO_HOME` environment variable to define where the TclPro installation exists.

For example, if you installed TclPro version 1.5 at `/u/tclpro1.5`, you must set the `SNPS_TCLPRO_HOME` variable to point to that directory. The IC Compiler tool uses this variable as a base for launching some of the TclPro tools. In addition, other Synopsys applications use this variable to link to the TclPro tools.

Running the Synopsys Syntax Checker

The Synopsys Syntax Checker for the IC Compiler tool is a standalone executable called `iccprocheck`. This executable is located at `$root/bin/iccprocheck`.

The following example is a script with errors that is tested in `iccprocheck`:

```
create_clock [get_port CLK] -p
create_clock [get_ports CLK] -p -12.2
sort_collection
set paths [get_timing_paths -nworst 10 -delay_type mx_fall -r]
my_report -from [sort_collection [sort_collection $a b] {b c d} -x]
foreach_in_collection x $objects {
    query_objects $x
    report_timing -through $x -thr $y -from a -from b -to z > r.out
}
all_fanout -from X -clock_tree
```

[Example 2-1](#) shows the resulting output from `iccprocheck`. Each line in the output shows where a syntax or semantic error was detected. The offending command is shown with a caret (^) below the character that begins the offending token.

Example 2-1 iccprocheck Output

```
Synopsys Tcl Syntax Checker - Version 1.0

Loading snps_tcl.pcx...
Loading icc.pcx...
scanning: /disk/scripts/ex1.tcl
checking: /disk/scripts/ex1.tcl
/disk/scripts/ex1.tcl:1 (warnUndefProc) undefined procedure: get_port
get_port CLK
^
/disk/scripts/ex1.tcl:1 (SnpsE-MisVal) Value not specified for
'create_clock -period'
create_clock [get_port CLK] -p
^
/disk/scripts/ex1.tcl:2 (SnpsE-BadRange) Value -12.2 for 'create_clock
-period' must be >= 0.000000
create_clock [get_ports CLK] -p -12.2
^
/disk/scripts/ex1.tcl:3 (SnpsE-MisReq) Missing required positional
options for sort_collection: collection criteria
sort_collection
^
/disk/scripts/ex1.tcl:4 (badKey) invalid keyword "mx_fall" must be:
max min min_max max_rise max_fall min_rise min_fall
get_timing_paths -nworst 10 -delay_type mx_fall -r
^
```

```
/disk/scripts/ex1.tcl:4 (SnpsE-AmbOpt) Ambiguous option
'get_timing_paths -r'
get_timing_paths -nworst 10 -delay_type mx_fall -r
^

/disk/scripts/ex1.tcl:5 (warnUndefProc) undefined procedure: my_report
my_report -from [sort_collection \
^
/disk/scripts/ex1.tcl:5 (SnpsE-UnkOpt) Unknown option
'sort_collection -x'
sort_collection [sort_collection $a b] {b c d} -x
^

/disk/scripts/ex1.tcl:9 (SnpsW-DupOver) Duplicate option
'report_timing -from' overrides previous value
report_timing -through $x -thr $y -from a -from b -to z > r.out
^

/disk/scripts/ex1.tcl:11 (SnpsE-Excl) Can only specify one of
these options for all_fanout: -from -clock_tree
all_fanout -from X -clock_tree
^
```

Limitations of the Synopsys Syntax Checker

The following limitations apply to the Synopsys Syntax Checker:

- Command abbreviations are not checked. Use of abbreviated commands show up as undefined procedures.
- Aliases created with the `alias` command are not expanded and show up as undefined procedures.
- A few checks done when the application is running might not be checked using the `snps_checker` environment. For example, some cases where one option requires another option are not checked.
- Script size is an issue with `snps_checker` and TclPro 1.3 and 1.5. Scripts of up to a few thousand lines can be reasonably checked, but beyond that, CPU time becomes a factor. Do not try to check extremely large scripts using `snps_checker`.
- The IC Compiler tool allows you to specify Verilog-style bus names on the command line without rigid quotation. An example of this is `A[0]`. This format with indexes from 0 to 255 is checked. The wildcard characters “*” and “%” are also checked. Other forms, including ranges such as `A[15:0]`, show up as undefined procedures, unless represented as `{A[15:0]}`.
- User-defined procedures enhanced with the `define_proc_attributes` command are not checked. Because such procedures are declared with the `args` argument, no semantic errors are reported.

Bytecode-Compiled Files

Bytecode-compiled files are created using the TclPro Compiler, procomp, and have the following advantages over ASCII scripts:

- They are efficient to load.
- They are secure because the content is not readable.
- The body of the compiled Tcl procedures is hidden.

The IC Compiler tool can load bytecode-compiled scripts. To load a bytecode-compiled file, you source it like you do other scripts. You do not need any files other than the application to load bytecode-compiled files.

TclPro Limitations

TclPro is not supported on RS/6000. However, bytecode-compiled files can be loaded into this application on all platforms, including RS/6000, and snps_checker is supported on all UNIX platforms.

Working With Licenses

You can view a list of the licenses you are currently using and a list of all licenses that are currently checked out. You can also check out additional licenses, queue licenses that are not currently available, and release licenses you no longer need. To learn how to determine what licenses are in use and know how to obtain and release licenses, see the following sections:

- [Listing the Licenses in Use](#)
- [Getting Licenses](#)
- [Enabling License Queuing](#)
- [Releasing Licenses](#)

Listing the Licenses in Use

To view the licenses that you currently have checked out, use the `list_licenses` command. For example,

```
icc_shell> list_licenses
```

```
Licenses in use:  
    Galaxy-Common  
    Galaxy-ICC  
1
```

To determine which licenses are already checked out, use the `license_users` command. For example,

```
icc_shell> license_users  
bill@eng1 Galaxy-Common, Galaxy-FP, Galaxy-ICC  
matt@eng2 Galaxy-Common, Galaxy-ICC, Milkyway-Interface  
2 users listed.  
1
```

Getting Licenses

When you invoke the IC Compiler tool, the Synopsys Common Licensing software automatically checks out the appropriate license. For example, if you open a Milkyway design library, Synopsys Common Licensing checks out the Milkyway-Interface license.

If you know the tools and interfaces you need, you can use the `get_license` command to check out those licenses. This ensures that each license is available when you are ready to use it. For example,

```
icc_shell> get_license Milkyway-Interface
```

After a license is checked out, it remains checked out until you release it or exit `icc_shell`.

Enabling License Queuing

The IC Compiler tool has a license queuing functionality that allows your application to wait for licenses to become available if all licenses are in use. To enable this functionality, set the `SNPSLMD_QUEUE` environment variable to `true`. The following message is displayed:

```
Information: License queuing is enabled. (ICCSH-001)
```

When you have enabled the license queuing functionality, you might run into a situation where two processes are waiting indefinitely for a license that the other process owns. To prevent such situations, set the `SNPS_MAX_WAITTIME` environment variable and the

The `SNPS_MAX_QUEUEETIME` environment variable. You can use these variables only if the `SNPSLMD_QUEUE` environment variable is set to `true`.

The `SNPS_MAX_WAITTIME` variable specifies the maximum wait time for the first key license that you require, such as the license for starting `icc_shell`. Consider the following scenario:

You have two IC Compiler packages, both of which are in use, and you are attempting to start a third `icc_shell` process. The queuing functionality places this last job in the queue for the specified wait time. The default wait time is 259,200 seconds (or 72 hours). If the license is still not available after the predefined time, you might see a message similar to the following:

```
Information: Timeout while waiting for feature 'Galaxy-ICC' (ICCSH-005)
```

The `SNPS_MAX_QUEUEETIME` variable specifies the maximum wait time for checking out subsequent licenses within the same `icc_shell` process. You use this variable after you have successfully checked out the first license to start `icc_shell`. Consider the following scenario:

You have already started the tool and are running a command that requires a Galaxy-FP license. The queuing functionality attempts to check out the license within the specified wait time. The default is 28,800 seconds (or eight hours). If the license is still not available after the predefined time, you might see a message similar to the following:

```
Information: Timeout while waiting for feature 'Galaxy-FP'. (ICCSH-005)
```

As you take your design through the synthesis flow, the queuing functionality might display other status messages, such as the following:

```
Information: Successfully checked out feature 'Galaxy-FP'. (ICCSH-002)
```

```
Information: Started queuing for feature 'Galaxy-FP'. (ICCSH-003)
```

```
Information: Still waiting for feature 'Galaxy-FP'. (ICCSH-004)
```

Releasing Licenses

To release a license that is checked out to you, use the `remove_license` command. For example,

```
icc_shell> remove_license Milkyway-Interface
```

Enabling Multicore Processing

Several IC Compiler functions support multicore processing, whether through distributed processing or multithreading. Multicore processing improves turnaround time by performing tasks in parallel much more quickly than if they were run in serial on a single core.

Note:

A single machine has one or more CPUs and each CPU has one or more cores. The total number of cores available for processing on a machine is the number of CPUs multiplied by the number of cores in each CPU.

When using multicore processing, you need one IC Compiler license for every four parallel tasks. For example, to run eight parallel tasks, you need two IC Compiler licenses.

The `set_host_options` command configures multicore processing in the IC Compiler tool.

To report the values set by the `set_host_options` command, use the `report_host_options` command. To remove the values set by the `set_host_options` command, use the `remove_host_options` command.

The following sections describe how to use the `set_host_options` command to configure both multithreading and distributed processing.

- [Configuring Multithreading](#)
 - [Configuring Distributed Processing](#)
-

Configuring Multithreading

Multithreading performs tasks in parallel by using multiple cores on the same machine, using a single process memory image. When using multithreading, each parallel task is called a thread. For the best performance during multithreading, you should limit the number of threads to the number of available cores, which is the number of CPUs in the machine times the number of cores per CPU.

To enable multithreading for those commands that support it, set the `set_host_options -max_cores` option to a value greater than one and less than or equal to the number of cores available on your machine. You can assign a name to the configuration by using the `-name` option; if you do not use this option, the tool assigns a name to the configuration.

When you enable multithreading, multithreaded commands create and use the specified number of threads, even if the number is more than the number of available cores. You must set an appropriate number of threads, so that the command does not try to use more resources than it has. Overthreading can reduce performance because the extra threads compete for resources. For best performance, do not run more than one thread per available core.

For example, if your machine has two CPUs and each CPU has three cores, specify six as the maximum number of threads:

```
icc_shell> set_host_options -max_cores 6
```

By default, the number of cores specified by the `-max_cores` option applies to all commands that support multithreading. You can use the variables shown in [Table 2-2](#) to limit the number of cores used for a specific function.

Table 2-2 Variables to Control Multicore Processing for Specific Functionality

Variable	Functionality
<code>extract_max_parallel_computations</code>	Extraction engine
<code>placer_max_parallel_computations</code>	Placement engine
<code>routeopt_max_parallel_computations</code>	<code>route_opt</code> command
<code>si_max_parallel_computations</code>	Signal integrity engine
<code>timing_max_parallel_computations</code>	Timing analysis engine
<code>zrt_max_parallel_computations</code>	Zroute

When you set these variables to a positive integer value, the tool limits the number of cores used for the associated function to the smaller of the value specified in the variable and the value specified in the `-max_cores` option of the `set_host_options` command.

Note:

For the signal integrity engine, the `si_max_parallel_computations` variable can only enable or disable multicore processing; it cannot modify the maximum number of cores. When this variable has a value of 0, multicore processing is enabled based on the `set_host_options` settings. When this variable has a value of 1, multicore processing is disabled and the signal integrity engine uses a single core regardless of the `set_host_options` settings.

Configuring Distributed Processing

Distributed processing performs tasks in parallel by using multiple machines; each process uses its own process memory image. When using distributed processing, each parallel task is called a process. For the best performance during distributed processing, you should limit the number of processes to the number of available cores, which is the sum of the number CPUs times the number of cores per CPU for each host machine.

The `set_host_options` command provides many options that can be used to configure distributed processing. At a minimum, you must specify the list of host machines (*list_of_hosts* argument); however, for the best performance, you should also specify the `-num_processes` option to limit the number of processes based on the number of available cores.

For example, to specify a distributed processing configuration that enables a maximum of four processes, with a maximum of two processes each on machineA and machineB, enter the following command:

```
icc_shell> set_host_options -name dp4 -num_processes 2 \
{machineA machineB}
```

To specify a distributed processing configuration that enables a maximum of four processes using the Load Sharing Facility (LSF), enter the following command:

```
icc_shell> set_host_options -name lsf4 -pool lsf -num_processes 4
```

To specify a distributed processing configuration that enables a maximum of four processes using the Oracle Grid Engine, enter the following command:

```
icc_shell> set_host_options -name grd4 -pool grd -num_processes 4
```

[Table 2-3](#) shows the command options for the `set_host_options` command.

Table 2-3 set_host_options Command Options

Command options	Description
<code>-name name</code>	Specifies a name for this configuration. You can use this name to modify, report, or remove this configuration. If you do not specify this option, the tool assigns a name.
<code>-max_cores count</code>	Specifies the maximum number of cores that can be used for multithreading. By default, a single core is used.

Table 2-3 set_host_options Command Options (Continued)

Command options	Description
<code>-num_processes count</code>	Specifies the maximum number of processes that can be used on each host machine for distributed processing. By default, there is no limit on the number of processes.
<code>-timeout timeout</code>	Specifies the job submission timeout value in seconds. If the jobs you submit do not become running jobs within the specified time, they are terminated to prevent them from hanging in the queuing system. Note: If you do not specify this option, the queuing system waits as long as needed for the jobs to start.
<code>-submit_command cmd</code>	Specifies the full path name of the LSF, Oracle Grid Engine, or custom job submission program. If you do not specify this option, <code>rsh</code> is used.
<code>-submit_options options</code>	Specifies options to be passed to the LSF, Oracle Grid Engine, <code>rsh</code> , or custom job submission program.
<code>-32bit</code>	Indicates that the program used by the distributed commands for executing their slave jobs is run in 32-bit mode. If you do not specify this option, 64-bit mode is used.
<code>-pool lsf grd</code>	Used by some commands to optimize their behavior for the specified type of queuing system.
<code>list_of_hosts</code>	Specifies the hosts used for distributed processing.

Exiting the IC Compiler Tool

You can end the session and exit the tool at any time. To exit the tool, use the `exit` or `quit` command.

Note:

When you exit the tool by using the `exit` or `quit` command, it does not save the open designs. For information about saving designs, see “[Saving a Design](#)” on page 3-60.

3

Preparing the Design

The IC Compiler tool uses a Milkyway design library to store your design and its associated library information. This chapter describes how to set up the libraries, create a Milkyway design library, read your design, and save the design in Milkyway format.

These steps are explained in the following sections:

- [Setting Up the Libraries](#)
- [Reading the Design](#)
- [Setting the Working Design](#)
- [Annotating the Floorplan Information](#)
- [Annotating the Scan Chain Information](#)
- [Specifying the Power Intent](#)
- [Validating the Design](#)
- [Creating Logical Power and Ground Connections](#)
- [Setting Up for Power Optimization](#)
- [Setting Up for Multicorner-Multimode Analysis and Optimization](#)
- [Preparing for Timing Analysis and RC Calculation](#)
- [Linking Designs](#)

- [Saving a Design](#)
- [Closing Designs](#)
- [Closing a Milkyway Design Library](#)
- [Archiving Designs](#)

Setting Up the Libraries

The IC Compiler tool requires both logic libraries and physical libraries. The following sections describe how to set up and validate these libraries:

- [Setting Up the Logic Libraries](#)
 - [Setting Up the Physical Libraries](#)
 - [Verifying the Libraries](#)
-

Setting Up the Logic Libraries

The IC Compiler tool uses logic libraries to provide timing and functionality information for all standard cells. In addition, logic libraries can provide timing information for hard macros, such as RAMs.

The tool supports logic libraries that use nonlinear delay models (NLDMs) and Composite Current Source (CCS) models and automatically selects the timing models to use, based on the contents of the logic libraries. If the logic libraries contain a mixture of both NLDM and CCS models, by default, the tool uses the CCS capacitance and timing data; you can control this by setting the `lib_pin_using_cap_from_ccs` and `lib_cell_using_delay_from_ccs` variables before loading the libraries.

Note:

When using CCS models, the tool might not use all of the available CCS data in some instances, such as during preroute optimization or on non-Arnoldi nets during postroute delay calculations. This is done to balance runtime against the required accuracy.

In each session, you must set up the logic libraries by defining the search path, link libraries, and target libraries.

- Search path

The search path lists, in order, the directories where the tool looks for logic library files that are specified without directory names.

Define the search path by setting the `search_path` variable. When the tool looks for a logic library file, it starts searching in the leftmost directory specified in the `search_path` variable and uses the first matching library file it finds.

- Link libraries

The link libraries are the logic libraries used to resolve cell references when the tool links the design.

Define the link libraries by setting the `link_library` variable. The first library specified in the `link_library` variable is the main library. If you are using an existing Milkyway

design library, ensure that the main library you specify is the same one that was used when the Milkyway design was saved.

The units used for time, voltage, current, resistance, capacitance, and power in the main library are used as the default units for the design. If your design or SDC file does not have unit settings, the tool uses these default settings.

By default, when the tool links the design by resolving the cell references, it first searches the local link libraries, which are defined in the `local_link_library` design attribute, and then the link libraries defined in the `link_library` variable. To disable searching of the local link libraries, set the `skip_local_link_library` variable to `true`.

- Target libraries

The target libraries are the logic libraries that the tool uses to perform physical optimization.

Define the target libraries by setting the `target_library` variable.

For more information about logic libraries, see the Library Compiler documentation.

Setting Up the Physical Libraries

The IC Compiler tool uses Milkyway reference libraries and technology files to obtain physical library information. The Milkyway reference libraries contain physical information about the standard cells and macro cells in your logic library. In addition, these reference libraries define the placement unit tile. The technology file provides technology-specific information, such as the name and characteristics of each metal layer.

The physical library information is stored in the Milkyway design library. For each cell, the Milkyway design library contains several views of the cell, which are used for different physical design tasks. Commonly used views include

- The layout (CEL) view
- The place and route (FRAM) view
- The metal fill (FILL) view
- The power and ground connectivity (CONN) view
- The error (ERR) view

If a Milkyway library does not already exist for your design, you need to create one and open it. If you already have a Milkyway design library, you must open it before working on your design.

This section describes the following tasks:

- [Creating a Milkyway Design Library](#)
- [Specifying the Milkyway Reference Libraries](#)
- [Opening a Milkyway Design Library](#)
- [Reporting a Milkyway Design Library](#)
- [Changing Physical Library Information](#)
- [Saving Physical Library Information](#)

For more information about Milkyway libraries, see the *Library Data Preparation for IC Compiler User Guide* and the *Milkyway Database Application Note*.

Creating a Milkyway Design Library

To create a Milkyway design library, use the `create_mw_lib` command (or choose File > Create Library).

At a minimum, you must specify the technology file that is used for the design, as well as the name of the Milkyway design library. For information about the technology file, see the *IC Compiler Technology File and Routing Rules Reference Manual*.

For example, to create a Milkyway design library named `my_design` in the current working directory and associate it with the `mytechfile.tf` technology file located in the `/usr/lib/tech` directory, enter the following command:

```
icc_shell> create_mw_lib my_design  
      -technology /usr/lib/tech/mytechfile.tf
```

If the new library uses the extended layer mode, run the `extend_mw_layers` command before the `create_mw_lib` command. The extended layer mode supports user-defined layer numbers from 1 to 4000, with layers 4001 through 4095 reserved for Milkyway system layers. For details, see “Extended Milkyway Layers” in the *Library Data Preparation for IC Compiler User Guide*.

Specifying the Milkyway Reference Libraries

Before working with the design, you must specify the Milkyway reference libraries used for the design. You can do this when you create the Milkyway design library with the `create_mw_lib` command or you can do it later by using the `set_mw_lib_reference` command.

You can specify the reference libraries on the command line or in a separate text file called a reference control file.

- To specify the reference libraries on the command line, use the `-mw_reference_library` option with the `create_mw_lib` or `set_mw_lib_reference` command.
- To specify the reference libraries in a reference control file, use the `-reference_control_file` option with the `create_mw_lib` or `set_mw_lib_reference` command.

When you specify the reference libraries, you can use either a relative or an absolute path to the library location. Note that when you use a relative path, the path is relative to the design library path, not the current directory. When the tool resolves cell references, it searches the libraries in the order they are specified, either on the command line or in the reference control file.

Note:

You can use the `set_mw_lib_reference` command to define the Milkyway reference libraries associated with a Milkyway design library only when the Milkyway design library is closed. If the design library is open, close it with the `close_mw_lib` command before defining the reference libraries.

For example, assume that the mydesign design library uses the refA and refB Milkyway libraries as reference libraries. The mydesign design library is in the current working directory and the reference libraries are in the /usr/lib/my_lib directory. To specify this relationship on the command line, use the following commands:

```
icc_shell> close_mw_lib
icc_shell> set_mw_lib_reference my_design \
    -mw_reference_library {/usr/lib/mw_lib/refA /usr/lib/mw_lib/refB}
```

You can also represent this relationship in a reference control file with the following content:

```
LIBRARY my_design
REFERENCE /usr/lib/mw_lib/refA
REFERENCE /usr/lib/mw_lib/refB
```

To specify this relationship using the reference control file, use the following commands:

```
icc_shell> close_mw_lib
icc_shell> set_mw_lib_reference my_design \
    -reference_control_file my_refs
```

If the my_design library contains cell A, the tool resolves this cell reference by first searching the refA library for the cell; if the cell is not found, it searches the refB library.

For more information about specifying the reference libraries, see the *Library Data Preparation for IC Compiler User Guide*.

Opening a Milkyway Design Library

To open an existing Milkyway design library, use the `open_mw_lib` command (or choose File > Open Library). When you open a Milkyway design library, the tool locks the library, which prevents other copies of it from being used and updated while you are working in it. You can open only one Milkyway design library at a time.

Note:

If you want to use the same Milkyway CEL version for multiple runs, copy the design by using the Copy Design dialog box (choose File > Copy Design) or the `copy_mw_cel` command.

To report the currently open Milkyway design library, use the `current_mw_lib` command.

If the Milkyway design library uses an older Milkyway database version (schema) and a database update is needed, the `open_mw_lib` command automatically updates the library data and issues a message about the conversion. For more information about Milkyway database versions and database updates, see the *Library Data Preparation for IC Compiler User Guide*.

If you specify the TLUPlus files before you open the Milkyway design library, IC Compiler performs consistency checking between the library files in the Milkyway design library and the TLUPlus files to ensure that you have a valid library setup. If you specify the TLUPlus files later, you must manually perform this validation by running the `check_tlu_plus_files` command. For more information, see “[Setting Up the TLUPlus Files](#)” on page 3-40.

Reporting a Milkyway Design Library

To report the reference libraries defined for the current Milkyway design library, use the `report_mw_lib` command with the `-mw_reference_library` option. [Example 3-1](#) shows a report on the reference libraries for the current Milkyway design library.

Example 3-1 Example of the report_mw_lib -mw_reference_library Command

```
icc_shell> open_mw_lib my_design
{my_design}
icc_shell> report_mw_lib -mw_reference_library
/usr/lib/mw_lib/refA
/usr/lib/mw_lib/refB
```

To report the information defined in the technology file associated with the current design library, use the `report_mw_lib` command with the `-unit_range` option. Note that you must specify the name of the design library to run this command. [Example 3-2](#) shows a report on the technology file for the `my_design` Milkyway design library.

Example 3-2 Example of the report_mw_lib -unit_range Command

```
icc_shell> report_mw_lib -unit_range my_design
```

Library: my_design	Tech. Attr.	Unit	Resolution	Min. Value	Max. Value
length	micron	1000	0.001	2147483.647	
time	ns	1000	0.001	2147483.647	
capacitance	pf	10000000	0.0000001	214.7483647	
resistance	kohm	10000000	0.0000001	214.7483647	
inductance	nh	100	0.01	21474836.47	
current	uA	1000	0.001	2147483.647	
voltage	V	1000	0.001	2147483.647	
power	pw	1000	0.001	2147483.647	

Layer: CP	Mask name: poly	Attribute	Minimum	Maximum
thickness		thickness	1.8e-01	1.8e-01
		unit resistance	0.0e+00	0.0e+00
		unit capacitance	0.0e+00	0.0e+00
		unit channel cap..	0.0e+00	0.0e+00
		unit sidewall cap.	0.0e+00	0.0e+00
		unit channel side cap.	0.0e+00	0.0e+00
		unit inductance	0.0e+00	0.0e+00
		height from substrate	0.0e+00	0.0e+00
		delta width	0.0e+00	0.0e+00
		min. area dimension	0.0e+00	
		min. object width	1.3e-01	
		min. object spacing	2.0e-01	
		max. wire length		0.0e+00
		max. RC seg. length		0.0e+00
		max. current density		0.0e+00
		intracap. dist. ratio		0.0e+00
		...		

To report the units used in the current design library, use the `report_units` command. [Example 3-3](#) shows a report on the units for the current Milkyway design library.

Example 3-3 Example of the report_units Command

```
icc_shell> open_mw_lib my_design
{my_design}
icc_shell> report_units

*****
Report : units
Design : my_design
Version: F-2011.09
Date   : Fri Jun 10 09:41:22 2011

*****
Units
-----
Time_unit      : 1.0e-09 Second(ns)
Capacitive_load_unit : 1.0e-12 Farad(pF)
Resistance_unit    : 1000 Ohm(kOhm)
Voltage_unit     : 1 Volt
Power_unit       : N/A
Current_unit     : 1.0e-06 Amp(uA)
1
```

To report the Milkyway database version (schema) and the revision information, use the `report_milkyway_version` command. To report the information for a specific cell, use the `-cell` option. To report the information for all cells, use the `-all` option.

Note:

All cells must be closed when running the `report_milkyway_version` command.

[Example 3-4](#) shows a report on the version information for the top cell in the current Milkyway design library.

Example 3-4 Example of the report_milkyway_version -cell Command

```
icc_shell> open_mw_lib my_design
{my_design}
...
icc_shell> report_milkyway_version -cell top

Cell Name          top.CEL;2
Data Model        6.0
Compatible Product E-2010.12 and F-2011.09
Creator           C-2009.06-ICC
Creation Time     Fri Sep 11 12:13:57 2009
Modifier          F-2011.09
Modification Time Tue Jun  7 16:48:29 2011
1
```

Changing Physical Library Information

You can change the technology file and Milkyway reference libraries associated with your design library.

Note:

You can change the physical library information only when the Milkyway design library is closed. If the design library is open, close it with the `close_mw_lib` command before changing the physical library information.

Changing the Technology File

To change the technology file associated with a design library, use the `set_mw_technology_file` command (or choose File > Set Technology File).

For example, to change the technology file associated with the `my_design` design library to `new_tech.tf`, enter the following command:

```
icc_shell> set_mw_technology_file -technology new_tech.tf my_design
```

Changing the Milkyway Reference Libraries

To change the Milkyway reference libraries associated with a design library, use the `set_mw_lib_reference` command (or choose File > Set Library Reference). In the same way as when you create a Milkyway design library, you can specify the Milkyway reference libraries either on the command line by using the `-mw_reference_library` option or in a reference control file by using the `-reference_control_file` option. For more information about specifying the reference libraries, see “[Specifying the Milkyway Reference Libraries](#)” on page 3-5.

For example, to change the Milkyway reference library associated with the `my_design` design library to `new_mw_ref`, enter the following command:

```
icc_shell> set_mw_lib_reference \
    -mw_reference_library {/usr/lib/mw_lib/new_mw_ref} my_design
```

Saving Physical Library Information

To save the technology file or reference control information in a file for later use, use the `write_mw_lib_files` command (or choose File > Export > Write Library File).

Note:

In a single invocation of the command, you can output only one type of file. To output both a technology file and a reference control file, you must run the command twice.

Writing the Technology File

To write out the technology file from an existing Milkyway design library, use the `write_mw_lib_files` command with the following syntax:

```
write_mw_lib_files
  -technology
  -output tech_file design_lib_file
```

For example, to save the technology file for the `my_design` design library in a file named `my_tech.tf` in the current working directory, enter the following command:

For example,

```
icc_shell> write_mw_lib_files -technology \
  -output my_tech.tf my_design
```

Writing the Reference Control File

To write out a reference control file from an existing Milkyway design library, use the `write_mw_lib_files` command with the following syntax:

```
write_mw_lib_files
  -reference_control_file
  -output reference_control_file design_lib_file
```

For example, to save the reference control file for the `my_design` design library in a file named `my.refs` in the current working directory, enter the following command:

```
icc_shell> write_mw_lib_files -reference_control_file \
  -output my.refs my_design
```

Verifying the Libraries

To achieve good results, you must have high-quality libraries. Before you process your design, you should use the `check_library` command to ensure that the logic libraries and physical libraries are correct and consistent.

By default, the `check_library` command performs consistency checking between the logic libraries specified in the `link_library` variable and the physical libraries referenced in the current Milkyway design library. If the `check_library` command reports any inconsistencies, you must fix them before you process your design.

For more information about performing library checking, see the *Library Data Preparation for IC Compiler User Guide*.

Reading the Design

The IC Compiler tool can read designs in Milkyway, .ddc, or ASCII (Verilog) format.

Regardless of the design format, you must perform the following tasks before reading in the design:

1. Set up the logic libraries, as described in “[Setting Up the Logic Libraries](#)” on page 3-3.

```
icc_shell> set_app_var search_path my_search_path  
icc_shell> set_app_var target_library my_library_list  
icc_shell> set_app_var link_library "* my_link_library_list"
```

2. Open the Milkyway design library associated with the design, as described in “[Opening a Milkyway Design Library](#)” on page 3-7.

```
icc_shell> open_mw_lib design_library_name
```

Note:

If you do not have a Milkyway design library, see “[Creating a Milkyway Design Library](#)” on page 3-5.

The following sections explain how to open designs:

- [Reading a Design in Milkyway Format](#)
- [Reading a Design in .ddc Format](#)
- [Reading a Design in ASCII Format](#)

Reading a Design in Milkyway Format

To open a design in Milkyway format,

1. Verify that you have defined the correct main library.

The unit settings in the Milkyway design must be consistent with the unit settings in the main library (the first library in the `link_library` definition). To see the main library unit settings, use the `report_lib` command. To see the Milkyway design library unit settings, use the `report_units` command.

To prevent problems with opening the Milkyway design, ensure that the main library is the same one that was used when the Milkyway design was saved.

2. Open the design by using the `open_mw_cel` command (or by clicking the  button on the File toolbar or choosing File > Open Design).

```
icc_shell> open_mw_cel design_name
```

Note:

If you use the GUI to open the design, you can open both the design library and the design from the Open Design dialog box.

3. If your design contains scan logic and the Milkyway design was saved in Design Compiler, annotate the scan chain information on the design.

Note:

If you have previously annotated the scan chain information on the design and have saved the Milkyway design in the IC Compiler tool, the SCANDEF information is already stored in the Milkyway design and you do not need to reannotate it.

For information about annotating the scan chain information, see “[Annotating the Scan Chain Information](#)” on page 3-21.

The Milkyway design contains the floorplan information and timing constraints previously set on the design, so you do not need to annotate this information separately.

If you get unit inconsistency errors when you try to open the Milkyway design, go back to step one and ensure that the main library is correctly defined.

When you open a Milkyway design, the tool

- Checks the database model (schema) version and automatically updates the database, if necessary

For more information about converting the database model, see the *Library Data Preparation for IC Compiler User Guide*.

- Checks the correctness and consistency of the netlist-related objects in the Milkyway design

To reduce runtime, the tool determines whether the design has previously passed these checks; if it has passed, the checks are skipped.

- Sets the design as the current design

- If the GUI is open, opens a new layout window and displays the design in the layout view

This is the primary design in the layout window; if you open multiple layout views in the same layout window, they all display the primary design. However, you can display other designs as overlays on the primary design.

Reading a Design in .ddc Format

To read a .ddc-format design,

1. Read the .ddc file for the design by using the `import_designs` command (or by choosing File > Import Designs in the GUI and selecting ddc as the input format).

```
icc_shell> import_designs -format ddc design.ddc
```

Note:

If you are using a bottom-up flow, you must have FRAM views for all blocks in the top-level design.

2. Annotate the floorplan information on the design.

For information about annotating the floorplan information, see “[Annotating the Floorplan Information](#)” on page 3-20.

The .ddc design contains the scan chain information and timing constraints previously set on the design, so you do not need to annotate this information separately.

After you read in the design, IC Compiler stores the design in Milkyway format for use in future sessions.

Reading a Design in ASCII Format

To read an ASCII-format design,

1. Read the Verilog netlist file for the design by using the `read_verilog` command (or by choosing File > Import > Read Verilog).

The Verilog netlist file is a structural or gate-level design in one file. For best results, the Verilog netlist should contain either a full power and ground network or no power and ground network.

```
icc_shell> read_verilog design.v
```

Note:

If you are using a bottom-up flow, you must have FRAM views for all blocks in the top-level design.

2. Uniquify the design by using the `uniquify_fp_mw_cel` command.

The Milkyway format does not support multiply instantiated designs. Before saving the design in Milkyway format, you must uniquify the design to remove multiple instances.

```
icc_shell> uniquify_fp_mw_cel
```

3. Annotate the floorplan information on the design.

For information about annotating the floorplan information, see “[Annotating the Floorplan Information](#)” on page 3-20.

4. If your design contains scan logic, annotate the scan chain information on the design.

For information about annotating the scan chain information, see “[Annotating the Scan Chain Information](#)” on page 3-21.

5. Read the timing constraints for the design by using the `read_sdc` command (or by choosing File > Import > Read SDC).

For information about setting timing constraints, see “[Setting Timing Constraints](#)” on page 3-47.

After you read in the design, the tool stores the design in Milkyway format for use in future sessions.

Handling Incomplete Data

During early design development, design data needs to be updated regularly. For example, a block recently updated by a block-level designer might have fewer pins than the same block that was used by a top-level designer.

By default, the tool fails to link the design if mismatches exist. To ignore mismatches and successfully link the design, set the `link_allow_design_mismatch` variable to `true`; this allows you to continue working on the design and gather useful information even in the presence of incomplete data.

To view the mismatches detected while linking a design, use the `report_design_mismatch` command. For example,

```
icc_shell> report_design_mismatch
*****
Report : design mismatches
Design  : my_design
Version: J-2014.09
Date   : Wed May 14 02:50:21 2014
*****  
  
Number of physical pin direction mismatch : 2  
  
Design      Object    Type      Mismatch
-----  
slow MXI2X4 libcell Ignored pin direction mismatch for 'S0' pin on
'MXI2X4' cell in /usr/design' physical library. Defaulting to the
direction specified in the 'slow' technology library.  
  
slow MX2X4  libcell Ignored pin direction mismatch for 'S0' pin on
'MX2X4' cell in /usr/design' physical library. Defaulting to the
direction specified in the 'slow' technology library.
```

Setting the Working Design

When you open a Milkyway design library, the tool sets it as the current design (the design you are actively working on). Most IC Compiler commands operate within the context of the current design. For example, if you use the `save_mw_cel` command without specifying the name of the design you want to save, the tool saves the current design.

Setting the Current Design

When multiple designs are open, you can reset the current design to a different design by using the `current_mw_cel` command (or by selecting a design name in the list on the Cells List toolbar in the main window).

Note:

If you open a design by using the View Settings panel to display it as an overlay on a layout view, it is not set as the current design.

Opening Designs in a Hierarchical Stack

Many large chip designs are organized as a hierarchy of physical designs. As you open and edit the various designs in a physical design hierarchy, you can use the `change_working_design` command to keep track of the parent-child relationships. This command also helps synchronize parent designs to edited child designs.

From a parent design, use the following command to open a child design:

```
icc_shell> change_working_design -push child_instance_name
```

This command has the same effect as opening the child design with the `open_mw_cel` command, but in addition, the parent-child relationship is maintained in a last-in, first out stack. By default, the child design is opened in write mode. To open the child design in read-only mode, use the `-readonly` option with the `change_working_design -push` command.

After you view or edit the child design, you can return to editing the parent design by using the following command:

```
icc_shell> change_working_design -pop
```

By default, this command synchronizes the changes made to the current design to its parent design, pops the current design out of the stack, and makes the parent design the current design. To discard the changes to the child design without synchronizing them to the parent design, use the `-discard` option with the `change_working_design -pop` command. Note that a pop operation does not save or close the child design. You can save all the parent and

child changes from the level of the parent design by using the `save_mw_cel -hierarchy` command.

Note:

If you open a design by using the `read_verilog` command, the top-level design is not set. Before you can use the `change_working_design -push` command, you need to set the current design by using the `current_mw_cel` command.

Traversing Multiple Levels

You can traverse the hierarchy multiple levels downward by using the `change_working_design -push` command multiple times. You can traverse the hierarchy back upward by using the `-pop` option multiple times. Each push operation causes the stack to grow by one entry and each pop operation causes the stack to shrink by one entry.

You can query the stack by using the `get_working_design_stack` command, as shown in the following example:

```
icc_shell> open_mw_cel Top
{Top}
icc_shell> change_working_design -push inst_celA
{Top celA}
icc_shell> change_working_design -push inst_celB
{Top celA celB}
...
icc_shell> get_working_design_stack
{Top celA celB}
...
icc_shell> change_working_design -pop
{Top celA}
icc_shell> change_working_design -pop
{Top}
```

Changing Between Different Stacks

When you open a design with the `open_mw_cel` command, it creates a new stack that contains the name of the design. If you use the `open_mw_cel` command to open multiple designs, it creates multiple stacks in memory. For example, consider the following sequence of commands:

```
icc_shell> open_mw_cel Top
{Top}
icc_shell> change_working_design -push inst_celA
{Top celA}
icc_shell> change_working_design -push inst_celB
{Top celA celB}
...
icc_shell> open_mw_cel inst_celX
{celX}
icc_shell> change_working_design -push inst_celY
{celX celY}
```

Now you have two stacks in memory. The current stack contains {celX celY}; the other stack contains {Top celA celB}.

You can switch the current working design back to the other stack by using the `change_working_design_stack` command.

```
icc_shell> change_working_design_stack Top
{Top celA celB}
```

In the `change_working_design_stack` command, you must reference the desired stack by the oldest, highest-level design name in the stack (`Top` in this example), but the command opens the newest, lowest-level design name in the stack (`celB` in this example).

Transforming Parent and Child Coordinates

The child and parent designs usually do not fall on the same coordinate system. In other words, the origin of a child design usually does not coincide with the origin of the parent design. To find out the coordinates of a location in a child design in terms of the parent design, or vice versa, use the `transform_coordinates` command.

For example, suppose you are editing a child design named `celA` that is instantiated in a parent design named `Top`, and you are editing an object at location `(100.0, 200.0)`, which is the location in terms of the child design. To compute the coordinates of this location in the parent design, use the following commands:

```
icc_shell> change_working_design -pop
{Top}
icc_shell> transform_coordinates -from_child celA {100.0 200.0}
{350.000 275.000}
```

The `transform_coordinates` command returns the coordinates of the location in the parent cell. You can similarly use the `-from_parent` option to transform the coordinates in a parent cell in terms of a child cell.

To use the `transform_coordinates` command, the parent and child designs must be contained within a stack created by a previous `change_working_design` command. A transform operation works only in a parent-child relationship and cannot span multiple levels of hierarchy. To transform coordinates between grandchild and grandparent cells, you need to use two separate transform operations with a `change_working_design` command between them.

Annotating the Floorplan Information

If a floorplan already exists for your design, use one of the following methods to annotate the floorplan information on the design:

- Reading the floorplan information from a DEF file
- Reading the floorplan information from a floorplan file
- Copying the floorplan information from another design

If a floorplan does not yet exist for your design, see the *IC Compiler Design Planning User Guide* for information about design planning.

Reading DEF Files

To read a DEF file, use the `read_def` command (or choose File > Import > Read DEF).

```
icc_shell> read_def design_name.def
```

By default, the `read_def` command is additive; it adds the floorplan information from the DEF file to the existing floorplan information in the design. To replace rather than add to existing floorplan information, use the `-no_incremental` option on the command line (or deselect Incremental in the GUI).

To analyze the input DEF files before annotating the floorplan information on the design, enable check-only mode by using the `-check_only` option. The check-only mode provides diagnostic information about the correctness and integrity of the DEF file. The check-only mode does not annotate any floorplan information into the Milkyway database.

```
icc_shell> read_def -check_only design_name.def
```

Reading Floorplan Files

A floorplan file is a file that you previously created by using the `write_floorplan` command (or by choosing Floorplan > Write Floorplan). This file is a Tcl script that contains the floorplan information.

Note:

Before reading the floorplan file, create the logical power and ground connections by using the `derive_pg_connection` command as described in “[Creating Logical Power and Ground Connections](#)” on page 3-26.

To read a floorplan file, use the `read_floorplan` command.

```
icc_shell> read_floorplan floorplan_file_name
```

For more information about creating and reading floorplan files, see the *IC Compiler Design Planning User Guide*.

Copying Floorplan Information

To copy floorplan information from the layout (CEL) view of one design in the current Milkyway design library to another, use the `copy_floorplan` command (or choose Floorplan > Copy Floorplan). Specify the design from which to copy the floorplan information by using the `-from` option (or by entering the name in the “From cell” box in the GUI).

```
icc_shell> copy_floorplan -from design1
```

For more information about copying floorplan information from another design, see the *IC Compiler Design Planning User Guide*.

Annotating the Scan Chain Information

The IC Compiler tool gets the scan chain information from a SCANDEF file, which specifies the scan chain components and their pins used for the scan path, along with reordering constraints. To get the best quality of results (QoR) for scan designs and enable repartitioning and reordering of the scan chains, you must annotate the scan chain information by loading the SCANDEF file.

Note:

If you read the design in .ddc format and the .ddc file contains embedded SCANDEF data, the scan chain information is already annotated on the design and you do not need to load the SCANDEF file. Loading the SCANDEF data from a .ddc file ensures that the netlist and the SCANDEF data are consistent. It also simplifies file management.

If your design contains scan chains, but you do not load the SCANDEF file, several IC Compiler commands issue an error message. To prevent this error message and continue with the flow, you must use the `-continue_on_missing_scandef` option with the command. When you use this option, the tool considers the scan chains during placement, which could result in reduced QoR.

For best results, you should use DFT Compiler for DFT insertion and Design Compiler topographical mode to generate the SCANDEF file (or the .ddc file with embedded SCANDEF data). For more information about DFT insertion, see the *DFT Compiler User Guide*. For more information about generating the SCANDEF file, see the *Design Compiler User Guide*.

Loading a SCANDEF File

To load a SCANDEF file, use the `read_def` command. To generate detailed information about errors and warnings that occur when you read the SCANDEF file, use the `-verbose` option.

```
icc_shell> read_def -verbose ./myscan.def
```

Note:

If you have previously annotated the scan chain information on the design and the SCANDEF data contains a chain with the same name as an existing scan chain, the existing chain definition is preserved. If you want to overwrite the existing chain definitions, you must first remove the SCANDEF data, as described in “[Removing SCANDEF Data](#)” on page 3-23.

After reading in the SCANDEF file, you are ready to run consistency checking.

Checking Scan Chain Consistency

The tool maintains the SCANDEF data that describes scan chain characteristics and constraints, as well as the scan-stitched netlist. The netlist and SCANDEF data must be consistent with each other. To validate their consistency, use the `check_scan_chain` command. By default, a summary of the consistency of all the chains is displayed. It includes the number of scan cells (sequential depth) in each chain.

Consistency checking is done on elements between the START and STOP points of the chain stubs. The check starts at the STOP point and proceeds backward. Each scan chain stub is given one of the following status values after the checks are complete:

- F

The scan chain failed validation, which means that the SCANDEF and the netlist are inconsistent for this chain.

- V

The scan chain is validated, which means that the SCANDEF and the netlist are consistent for this chain.

Only validated scan chain stubs are optimized.

The following items are not checked by the `check_scan_chain` command:

- MAXBITS and scan chain lengths
- PARTITION labels

You can perform further validation by using the Formality equivalency checker to ensure the pre-layout netlist is equivalent to the post-layout netlist, with test signals de-asserted or not compared.

Removing SCANDEF Data

To remove all SCANDEF data that was previously annotated on the design, use the `remove_scan_def` command.

```
icc_shell> remove_scan_def
```

Specifying the Power Intent

The IEEE 1801 standard, also known as Unified Power Format (UPF), establishes a set of commands used to specify the low-power design intent for electronic systems. Using UPF commands, you can specify the supply network, switches, isolation, retention, and other aspects relevant to power management of a chip design. The same set of low-power design specification commands is to be used throughout the design, analysis, verification, and implementation flow. Synopsys tools are designed to follow the official UPF standard.

The UPF language provides a way to specify the power requirements of a design, but without specifying explicitly how those requirements are implemented. The language specifies how to create a power supply network for each design element, the behavior of supply nets with respect to each other, and how the logic functionality is extended to support dynamic power switching to design elements. It does not contain any placement or routing information.

In the UPF language, a *power domain* is a defined group of elements in the logic hierarchy that share a common set of power supply needs. By default, all logic elements in a power domain use the same primary supply and primary ground. Other power supplies can optionally be defined for a power domain as well. A power domain is typically implemented as a contiguous *voltage area* in the physical chip layout, although this is not a requirement of the language.

Each power domain has a *scope* and an *extent*. The *scope* is the level of logic hierarchy where the power domain exists. The *extent* is the set of logic elements that belong to the power domain and share the same power supply needs. In other words, the scope is the hierarchical level where the power domain exists, whereas the extent is what is contained within the power domain.

Each scope or hierarchical level in the design has *supply nets* and *supply ports*. A *supply net* is a conductor that carries a supply voltage or ground throughout a given power domain. A supply net that spans more than one power domain is said to be “reused” in multiple domains. A *supply port* is a power supply connection point between two adjacent levels of the design hierarchy, between parent and child blocks of the hierarchy. A supply net that crosses from one level of the design hierarchy to the next must pass through a supply port.

A *power switch* (or simply *switch*) is a device that turns on and turns off power for a supply net. A switch has an input supply net, an output supply net that can be switched on or off, and at least one input signal to control switching. The switch can optionally have multiple input control signals and one or more output acknowledge signals. A *power state table* lists the allowed combinations of voltage values and states of the power switches for all power domains in the design.

Where a logic signal leaves one power domain and enters another at a substantially different supply voltage, a *level-shifter* cell must be present to convert the signal from the voltage swing of the first domain to that of the second domain.

Where a logic signal leaves a power domain and enters a different power domain, an *isolation* cell must be present to generate a known logic value during shutdown. If the voltage levels of the two domains are substantially different, the interface cell must perform both level shifting when the domain is powered up and isolation when the domain is powered down. A cell that can perform both functions is called an *enable level shifter*.

In a power domain that has power switching, any registers that are to retain data during shutdown must be implemented as *retention registers*. A retention register has a separate, always-on supply net, sometimes called the backup supply, which keeps the data stable in while the primary supply of the domain is shut down.

For details about specifying the power intent, see [Chapter 12, “Multivoltage Design Flow”](#) and the “Power Intent Specification” chapter in the *Synopsys Multivoltage Flow User Guide*.

Validating the Design

Before continuing, you should validate the floorplan information and the integrity of the design database.

Validating the Floorplan Information

To validate the floorplan information in your design, run the `check_physical_design` command. When you run this command, you must specify the current stage of your design to ensure that the proper physical design checks are run.

- Before running the `place_opt` command, run the `check_physical_design` command with the `-stage pre_place_opt` option.

```
icc_shell> check_physical_design -stage pre_place_opt
```

- Before running the `clock_opt` command, run the `check_physical_design` command with the `-stage pre_clock_opt` option.

```
icc_shell> check_physical_design -stage pre_clock_opt
```

- Before running the `route_opt` command, run the `check_physical_design` command with the `-stage pre_route_opt` option.

```
icc_shell> check_physical_design -stage pre_route_opt
```

Validating the Integrity of the Design Database

To validate the integrity of the design database, run the `check_database` command. By default, the `check_database` command verifies that

- The power and ground network is consistent
- There are no logical connections to physical-only cells
- The UPF data is consistent, if it exists
- The design has been unqualified and contains hierarchy preservation data

To perform only the logical checks, use the `-netlist` option. To perform only the physical checks, use the `-physical` option. You can increase the verbosity of the generated messages by setting the `-verbosity` option to `medium` or `high` (the default is `low`).

You should run the `check_database` command

- After reading a design in ASCII format using the `read_def` command
- Before running application commands, such as `change_names`, `create_block_abstraction`, `place_opt`, `clock_opt`, and `route_opt`. This can help save runtime in the case of a data error
- Before using the `write_verilog` command. For information about the `write_verilog` command, see “[Saving the Design in ASCII Format](#)” on page 3-63

Creating Logical Power and Ground Connections

After you read in the design, you must ensure that there are logical connections between the power and ground nets and the power, ground, and tie-off pins on the cells in your design. If your design does not already have these connections, use the `derive_pg_connection` command (or Preroute > Derive PG Connection in the GUI) to create them. This command creates the logical power and ground connections for leaf cells, hierarchical cells, and physical-only cells in both single-voltage and multivoltage designs.

To show the logical power and ground connections made by the `derive_pg_connection` command, use the `report_cell_physical -connections` command.

You should rerun the `derive_pg_connection` command whenever the power and ground connections have changed, such as

- After design planning
- After using logic ECO to modify the design
- After chip-finishing tasks

To ensure correct power and ground information in the Verilog netlist, you should also run the `derive_pg_connection` command before writing a Verilog netlist.

Creating the Connections for Single-Voltage Designs

To create or update the logical power and ground connections for a single-voltage design,

1. Connect the power and ground pins to the power and ground nets.

To connect all power and ground pins for a single-voltage design, specify the power and ground net names when you run the `derive_pg_connection` command.

```
icc_shell> derive_pg_connection -power_net VDD -ground_net VSS
```

To connect specific power and ground pins for a single-voltage design, you must also specify the power and ground net and pin names when you run the `derive_pg_connection` command.

```
icc_shell> derive_pg_connection -power_net VDD -power_pin vdd \  
      -ground_net VSS -ground_pin vss
```

If the design does not yet have power and ground ports, use the `-create_ports top` option to create these ports.

If the design already has logical power and ground connections but you want to regenerate these connections, use the `remove_pg_network -top` command to remove the existing power and ground network before you run the `derive_pg_connection` command.

By default, the `derive_pg_connection` command does not change existing power and ground pin connections. To reconnect the power and ground pins, use the `-reconnect` option when you run the `derive_pg_connection` command.

2. Connect the tie-off pins to the power and ground nets.

If the design uses tie-off cells for tie-off connections, you do not need to perform this step; the `place_opt` or `psynopt` command performs tie-cell insertion and connects the tie-off cells to the tie-off pins.

If the design does not use tie-off cells, use the `derive_pg_connection` command with the `-tie` option to connect the tie-off pins.

```
icc_shell> derive_pg_connection -power_net VDD -ground_net VSS -tie
```

In a hierarchical design, the tie-off connections are mapped to actual power and ground nets in the same hierarchy.

By default, the `derive_pg_connection -tie` command does not change any existing tie-off connections; it only incrementally connects unconnected tie-off pins. To change existing tie-off connections, use the `recover_tie_connection` command and then use

the `derive_pg_connection -tie` command with the proper options to remap the tie-off nets to power or ground nets. Note that the `-reconnect` option of the `derive_pg_connection` command does not reconnect tie-off nets.

Creating the Connections for Multivoltage Designs

This section describes how to create the logical power and ground connections for multivoltage designs with UPF specifications. If the power intent for your multivoltage design is not specified using UPF, see [SolvNet article 031133, “Power, Ground, and Tie-Off Editing Application Note,”](#) for information about using the `derive_pg_connection` command in non-UPF mode.

To create or update the logical power and ground connections for a multivoltage design,

1. Create the supply nets, if they do not already exist.

To create the supply nets based on the UPF specification, run the `derive_pg_connection` command with the `-create_nets` option.

```
icc_shell> derive_pg_connection -create_nets
```

When you use the `-create_nets` option, the `derive_pg_connection` command checks for the following conflicts and potential problems in the power and ground network:

- Incomplete power and ground network with a UPF conflict
- Signal tie-net name that is inconsistent with the UPF supply net name
- Signal net with a power or ground name but without a power or ground source
- Signal net with a power or ground name at a middle hierarchy level connected to a signal at the top level

The `derive_pg_connection -create_nets` command does not create the supply nets for the design unless all conflicts are resolved. To report the network changes necessary to make the power and ground network compatible with the UPF power intent, run the `derive_pg_connection -resolve_conflict` command.

2. Connect the power and ground pins to the power and ground nets.

To connect the power and ground pins for a multivoltage design, run the `derive_pg_connection` command.

```
icc_shell> derive_pg_connection
```

The `derive_pg_connection` command derives all power and ground nets, power and ground ports, and connections from the UPF specification.

By default, the `derive_pg_connection` command does not change existing power and ground pin connections. To override the existing power connections and use the power

intent defined in the UPF file, use the `-reconnect` option when you run the `derive_pg_connection` command.

3. Connect the tie-off pins to the power and ground nets.

If the design uses tie-off cells for tie-off connections, you do not need to perform this step; the `place_opt` or `psynopt` command performs tie-cell insertion and connects the tie-off cells to the tie-off pins.

If the design does not use tie-off cells, use the `derive_pg_connection` command with the `-tie` option to connect the tie-off pins.

```
icc_shell> derive_pg_connection -tie
```

By default, the `derive_pg_connection -tie` command does not change any existing tie-off connections; it only incrementally connects unconnected tie-off pins. To change existing tie-off connections, use the `recover_tie_connection` command and then use the `derive_pg_connection -tie` command to remap the tie-off nets to power and ground nets.

Setting Up for Power Optimization

If you plan to perform power optimization on your design, you must

- Prepare the logic libraries for leakage-power optimization
- Annotate the switching activity on your design
- Set the multiple-threshold-voltage constraint

Preparing Logic Libraries for Leakage-Power Optimization

The IC Compiler tool uses library cell attributes to identify the cells to be used for leakage-power optimization.

- Adaptive leakage-power optimization uses multiple-threshold-voltage cells to minimize the leakage power. The tool uses the `default_threshold_voltage_group` library attribute and the `threshold_voltage_group` library cell attribute to identify the high-threshold-voltage and low-threshold-voltage cells.
For more information about adaptive leakage-power optimization, see “[Performing Power Optimization](#)” on page 4-35.
- In addition to these attributes, final stage leakage-power recovery uses the `cell_footprint` library cell attribute to identify candidate cells for leakage-power optimization.
For more information about final stage leakage recovery, see “[Performing Final Stage Leakage-Power Recovery](#)” on page 6-107.

The following sections describe how to set and query these attributes.

Identifying Multiple-Threshold-Voltage Cells

If your target library does not support multiple-threshold-voltage cells and the associated attributes, you can use the `set_attribute` command to identify the multiple-threshold-voltage cells.

The following example shows how you set the `default_threshold_voltage_group` library attribute on the `high_vt_library` and `low_vt_library` library files, which contain cells of the high-threshold-voltage group and low-threshold-voltage group respectively in separate library files.

```
set HVT_lib "high_vt_library"
set LVT_lib "low_vt_library"
set_attribute [get_libs $HVT_lib] default_threshold_voltage_group HVT
set_attribute [get_libs $LVT_lib] default_threshold_voltage_group LVT
```

If you have multiple-threshold-voltage cells in the same library file, you can set the `default_threshold_voltage_group` library attribute and `threshold_voltage_group` library cell attributes as shown in the following example:

```
set all_vt_lib "multiple_vt_lib"
set_attribute [get_libs $all_vt_lib] default_threshold_voltage_group HVT
set_attribute [get_lib_cells ${all_vt_lib}/FAST*] \
    threshold_voltage_group LVT
set_attribute [get_lib_cells ${all_vt_lib}/MEM*] \
    threshold_voltage_group MEM
```

Analyzing the Multiple-Threshold-Voltage Cell Characteristics

To report the leakage power and timing characteristics of library cells that belong to different threshold-voltage groups, use the `analyze_library -multi_vth` command. This command reports the geometric average of the leakage power per area and timing for each threshold-voltage group, normalized by the values for the threshold-voltage group with the smallest leakage power per area. It also reports the number of don't use cells and the total number of cells for each threshold-voltage group.

By default, the command analyzes the cells in all the target libraries. However, you can limit it to specific libraries by specifying the corresponding .db file names, as shown in the following example:

```
icc_shell> analyze_library -multi_vth {sclmd_40c.db scsxd_40c.db}
*****
          Multi-VT Library Analysis Report
*****
          Vth Group/Library Name      Avg. Leakage      Avg.
          (don't use cells/total cells)    per Area      Timing
*****
          lmd (10/661)                1.00           1.00 (Baseline)
          sxd (10/661)                1.14           1.62
*****
```

Preparing Logic Libraries for Final Stage Leakage-Power Recovery

Like adaptive leakage-power optimization, final stage leakage-power recovery requires that the multiple-threshold-voltage cells are identified in the logic library. In addition to this requirement, final stage leakage-power recovery requires that the `cell_footprint` attribute exists on the library cells. If the `cell_footprint` attribute is not present, you can use the `set_attribute` command to set this attribute.

Note that the `cell_footprint` attribute of the library cells must be consistent with the number of threshold voltage groups in the logic libraries. When you run final stage leakage-power recovery, the tool issues the FOPT-028 message to report the number of threshold-voltage groups being used.

Information: Optimizing leakage power with 3 voltage thresholds (FOPT-028).

Annotating the Switching Activity

You obtain better power savings if you annotate switching activity on the design before you perform power optimization. You can annotate the switching activity in the following ways:

- Reading the switching activity file

You use the `read_saif` command to read the switching activity file. The `read_saif` command has the following syntax:

```
read_saif -input saif_file -instance_name path
```

- Specifying the switching activity

You can specify the switching activity by using the `set_switching_activity` command.

If you do not specify the switching activity, the tool applies the default toggle rate to the primary inputs and black box outputs and then propagates it throughout the design.

Specifying the Multiple-Threshold-Voltage Constraint

When you have multiple-threshold-voltage cells, supported either in the logic library or by setting appropriate attributes on the library cells, the tool chooses cells that belong to the appropriate threshold-voltage groups to minimize the leakage power, without violating the timing and design rule constraints. To set the multiple-threshold-voltage constraint, use the `set_multi_vth_constraint` command. This command also allows you to specify whether this constraint should have higher or lower priority than the timing constraint.

Use the `-lvth_percentage` option to specify the percentage value. The value can be a floating-point number between 0 and 100. This number represents the maximum percentage of the low-threshold-voltage cells in the synthesized design.

The `-type` option specifies whether the constraint is hard or soft. When you specify `-type hard`, the tool tries to meet this constraint even if this results in timing degradation. If you specify `-type soft`, the tool tries to meet this constraint only if that does not degrade the timing QoR.

While calculating the percentage of low-threshold-voltage cells in the design, the tool does not consider the black box cells. To let the tool consider the black box cells in the percentage calculation, use the `-include_blackboxes` option.

In the following example, the maximum percentage of low-threshold-voltage cells in the design is set to 15 percent. While trying to meet this constraint, the timing constraint is not compromised.

```
icc_shell> set_multi_vth_constraint -lvth_groups {lvt svt} \
    -lvth_percentage 15 -include_blackboxes -type soft
```

When you perform leakage-power optimization by using the `place_opt -power`, `place_opt_feasibility -power`, `psynopt -power`, `psynopt -only_power`, `clock_opt -power`, `clock_opt_feasibility -power`, `route_opt -power`, or `route_opt -only_power_recovery` command, the tool tries to meet the multiple-threshold-voltage constraints you set by using the `set_multi_vth_constraint` command. If you do not set multiple-threshold-voltage constraints, the tool optimizes for leakage power based on the leakage values of the library cells.

To see the percentage of cells for each threshold-voltage group in the design, use the `report_threshold_voltage_group` command.

Setting Up for Multicorner-Multimode Analysis and Optimization

Designs are often required to operate under multiple operating conditions (“corners”) and in multiple modes. Such designs are referred to as multicorner, multimode, or multicorner-multimode designs. These designs can involve numerous operating condition corners and many modes. The IC Compiler tool uses a truly concurrent, multi-scenario method to analyze and optimize these designs across all design corners and modes of operation.

A *scenario* is a combination of modal constraints and corner specifications. In multicorner-multimode designs, the tool uses a scenario or a set of scenarios as the unit for analysis and optimization. The *current scenario* is the focus scenario; when you set modal constraints or corner specifications, these typically apply to the current scenario. The *active scenarios* are the set of scenarios used for timing analysis and optimization.

The constraints, such as the clock specifications and false paths, describe the different modes of operation of the design. The operating conditions, extraction specifications, timing derating factors, and so on, form the corner specification of the multicorner-multimode design. Some timing constraints can be part of both the mode specification and the corner specification.

Timing analysis is carried out on all scenarios concurrently, and cost is measured across all scenarios for timing and design rule constraints. As a result, the timing and constraint reports show worst-case timing across all scenarios.

Concurrent multicorner-multimode optimization works on the worst violations across all scenarios, eliminating the convergence problems observed in sequential approaches. Optimization is performed for design rule constraints, setup, and hold, and is supported for both preroute and postroute designs.

The following topics describe how to prepare a multicorner-multimode design for analysis and optimization:

- [Defining Scenarios](#)
 - [Checking the Scenario Definitions](#)
 - [Managing the Scenarios](#)
-

Defining Scenarios

To define a scenario,

1. Create the scenario.

Use the `create_scenario` command to create a scenario. When you create a scenario, that scenario becomes the current scenario and is considered an active scenario.

You can create scenarios either before or after reading the design. If you create scenarios before reading the design, all the scenario definitions are applied to the design being read. If you create scenarios after reading the design, when you create the first scenario, all previous scenario-specific constraints are removed from the design and this warning is issued:

```
icc_shell> create_scenario s1
Warning: Any existing scenario-specific constraints are
discarded. (MV-020)
Current scenario is: s1
```

2. Specify the TLUPlus libraries, operating conditions, and constraints that apply to the scenario.

In general, when you specify these items, they apply to the current scenario. To determine the current scenario, use the `current_scenario` command without an argument. To change the current scenario, use the `current_scenario` command to specify the name of the current scenario. For example,

```
icc_shell> current_scenario
Current scenario is: s2
icc_shell> current_scenario s1
Current scenario is: s1
```

For more information about specifying the TLUPlus libraries, operating conditions, and constraints for a scenario, see “[Preparing for Timing Analysis and RC Calculation](#)” on [page 3-40](#).

When you save a design in Milkyway format after defining scenarios for the design, by default, all the defined scenarios are saved with the design. For more information about saving designs in Milkyway format, see “[Saving the Design in Milkyway Format](#)” on page 3-60.

Checking the Scenario Definitions

To perform consistency checks between scenarios, use the `check_scenarios` command. This command checks for scenario-specific information such as specifications of TLUPlus files, operating conditions, libraries, and clocks. By default, the output is saved in the current directory in HTML format. Use the `-output` option to specify a different location for the output file. Use the `-display` option to see the output in a Web browser. When you use this option, the tool displays the output in the browser defined by the `gui_online_browser` variable.

Managing the Scenarios

The following sections describe how to manage scenarios:

- [Selecting the Scenarios to Use](#)
- [Reporting the Scenarios](#)
- [Removing Scenarios](#)

Selecting the Scenarios to Use

By default, all scenarios that are applied to the current design are used for timing analysis and optimization. To reduce memory and runtime, you can modify the scenarios usage by

- [Setting the Active Scenarios](#)
- [Using Scenario Reduction](#)
- [Enabling and Disabling Scenarios for Specific Tasks](#)

Setting the Active Scenarios

An active scenario is a scenario that is used for the enabled optimization and analysis tasks. By default, all scenarios are active scenarios; however, some scenarios might not be relevant for a specific optimization or analysis context. To specify the set of active scenarios, use the `set_active_scenarios` command.

Using Scenario Reduction

During concurrent analysis and optimization, you can significantly reduce memory usage and runtime by limiting the number of active scenarios to those that are essential or dominant. A dominant scenario has the worst slack among all the scenarios for at least one of its constrained objects. Constrained objects can include delay constraints associated with a pin, the design rule constraints for a net, leakage power, and so on. Any scenario for which none of its constrained objects has the worst slack value is not a dominant scenario. The process of determining the dominant scenarios is called *scenario reduction*.

Scenario reduction analysis is always based on the current state of the design—that is, the design's current placement, routing, and clock tree state. In general, restricting concurrent analysis and optimization to a subset of dominant scenarios does not lead to significant QoR degradation.

To determine the dominant scenarios, use the `get_dominant_scenarios` command. By default, the `get_dominant_scenarios` command analyzes all active scenarios. Use the `get_dominant_scenarios -scenarios` command to restrict the set of scenarios considered during reduction. The list of scenarios you specify can include both active and inactive scenarios. The command identifies the dominant set from the list of specified scenarios. When you use the `get_dominant_scenarios -apply` command, the dominant scenarios become active and all other scenarios are inactive. The `-apply` option improves runtime if the current scenario before running this command is not in the scenario list returned by the `get_dominant_scenarios` command.

Note:

The `get_dominant_scenarios` command only supports designs that are already placed.

To improve turnaround time without adversely affecting QoR, you can run the `get_dominant_scenarios` command across multiple cores.

- To run the `get_dominant_scenarios` command in parallel on multiple cores on the same machine, use the `set_host_options -max_cores` command to specify the number of cores available on your machine.
- To run the `get_dominant_scenarios` command in parallel across several machines,
 1. Use the `set_host_options` command to configure the distributed processing.
 2. Use the `-distributed` option of the `get_dominant_scenarios` command to run the command in distributed mode.

For more information about configuring multicore processing, see “[Enabling Multicore Processing](#)” on page 2-12.

Enabling and Disabling Scenarios for Specific Tasks

The tasks listed in [Table 3-1](#) use only those scenarios enabled for that task, rather than using all active scenarios. To perform a specific task, at least one active scenario must be enabled for that task; otherwise, the tool does not perform the task.

By default, scenarios are enabled only for delay optimization and analysis (both maximum and minimum). Note that enabling a scenario for maximum delay optimization and analysis also enables it for maximum transition and maximum capacitance design rule fixing and analysis.

To enable or disable a scenario for specific tasks, use the `set_scenario_options` command with one or more of the options shown in [Table 3-1](#). By default, the options are applied to the current scenario. To apply the options to specific scenarios, use the `-scenarios` option.

Table 3-1 Scenario-Specific Tasks

Task	Option	Default
Timing optimization and analysis		
Maximum delay optimization and analysis (also enables the specified scenarios for maximum transition and maximum capacitance design rule fixing and analysis)	<code>-setup</code>	true
Minimum delay optimization and analysis	<code>-hold</code>	true
Power optimization and analysis		
Leakage-power optimization and analysis	<code>-leakage_power</code>	false
<ul style="list-style-type: none"> A scenario enabled for leakage-power optimization is referred to as a leakage scenario In general, you can enable multiple leakage scenarios; however, final stage leakage-power recovery, which is invoked by running the <code>focal_opt -power</code> command, requires a single leakage scenario. For more information about final stage leakage-power recovery, see “Performing Final Stage Leakage-Power Recovery” on page 6-107. 		
Dynamic-power optimization and analysis	<code>-dynamic_power</code>	false
<ul style="list-style-type: none"> To use a scenario for dynamic power optimization and analysis, maximum delay optimization (<code>-setup</code>) must also be enabled for that scenario. Disabling maximum delay optimization for a scenario automatically disables dynamic-power optimization and analysis for that scenario. 		

Table 3-1 Scenario-Specific Tasks (Continued)

Task	Option	Default
Clock tree synthesis and optimization		
Clock tree synthesis	-cts_mode	false
<ul style="list-style-type: none"> A scenario enabled for clock tree synthesis is referred to as a clock tree synthesis scenario During clock tree synthesis, all clock tree synthesis scenarios are activated regardless of their current state. After clock tree synthesis, they are returned to their current state (active or inactive). 		
Clock tree optimization	-cts_corner (Valid values are none, min, max, or min_max)	none
<ul style="list-style-type: none"> Clock tree optimization uses the specified corner of the clock tree synthesis scenarios to perform clock tree optimization If you do not specify this option for a clock tree synthesis scenario, the tool uses the maximum corner. 		

To report the scenario options, use the `report_scenario_options` command.

Reporting the Scenarios

The following sections describe the commands used to check and report scenarios.

To display all the defined scenarios, use the `all_scenarios` command.

```
icc_shell> all_scenarios
s1 s2
```

To report all the defined scenarios, use the `report_scenarios` command.

```
icc_shell> report_scenarios
```

The following example shows a report generated by the `report_scenarios` command:

```
*****
Report : scenarios
Design : DESIGN1
scenario(s) : SCN1
Version: A-2007.12
Date   : Thu Nov 08 20:55:59 2007
*****

All scenarios (Total=4): SCN1 SCN2 SCN3 SCN4
All Active scenarios (Total=1): SCN1
Current scenario      : SCN1
CTS Scenario          : SCN3

Scenario #0: SCN1 is active.
Scenario options:
Has timing derate: No
Library(s) Used:
  technology library name (File: library.db)

Operating condition(s) Used:
  Analysis Type      : bc_wc
  Max Operating Condition: library:WCCOM
  Max Process        : 1.00
  Max Voltage        : 1.08
  Max Temperature    : 125.00
  Min Operating Condition: library:BCCOM
  Min Process        : 1.00
  Min Voltage        : 1.32
  Min Temperature    : 0.00

Tlu Plus Files Used:
  Max TLU+ file: tlu_plus_file.tf
  Tech2ITF mapping file: tf2itf.map

Number of leakage-only scenario(s): 0
```

To display the current scenario, use the `current_scenario` command without an argument.

```
icc_shell> current_scenario
Current scenario is: s2
```

To display the currently active scenarios, use the `all_active_scenarios` command.

```
icc_shell> all_active_scenarios
s1 s2
```

Removing Scenarios

Use the `remove_scenario` command to remove the specified scenarios from memory. To remove all scenarios use the `-all` option. You can use this command to control the scenarios applied to, or saved with a design.

Preparing for Timing Analysis and RC Calculation

The IC Compiler tool provides RC calculation technology and timing analysis capabilities for both preroute and postroute data. The following sections describe the tasks you should complete before you perform RC calculation and timing analysis:

- [Setting Up the TLUPlus Files](#)
 - [Specifying the Operating Conditions](#)
 - [Setting Timing Constraints](#)
 - [Selecting the Delay Calculation Method \(optional\)](#)
 - [Back-Annotating Delay or Parasitic Data \(optional\)](#)
-

Setting Up the TLUPlus Files

TLUPlus is a binary table format that stores the RC coefficients. The TLUPlus models enable accurate RC extraction results by including the effects of width, space, density, and temperature on the resistance coefficients. For details about modeling these effects in the Interconnect Technology Format (ITF) file, see the StarRC documentation.

To use TLUPlus models for RC estimation and extraction, specify the following files:

- The mapping file, which maps the metal and via layer names in the Milkyway technology file to the layer names in the ITF file

You must specify the mapping file if the layer names differ between the technology file and the ITF file or if the technology file defines layers that are not in the ITF file. If the names are the same in the technology file and the ITF file, you do not need to specify this file.

The mapping file has the following format:

```
conducting_layers
tech_file_layer_name ITF_file_layer_name
...
via_layers
tech_file_layer_name ITF_file_layer_name
...
[remove_layers
tech_file_layer_name
...]
```

Specify the metal layer name mapping in the `conducting_layers` section and the via layer name mapping in the `via_layers` section. If the technology file defines metal or via layers that are not used for routing and are not used in the ITF file, specify these layers in the optional `remove_layers` section.

- The maximum TLUPlus model file
- (Optional) The minimum TLUPlus model file

You must specify the minimum TLUPlus file if the minimum and maximum operating conditions are different and the TLUPlus models have derating factors. In that case, you must specify the minimum file even if it is the same as the maximum file.

You specify these files by using the `set_tlu_plus_files` command (or by choosing File > Set TLU+ in the GUI). You can specify TLUPlus file names with or without their full paths. If you do not include the paths, the tool uses the search paths defined with the `search_path` variable. For example,

```
icc_shell> set_tlu_plus_files \
    -tech2itf_map ./path/map_file_name.map \
    -max_tluplus ./path/worst_settings.tlup \
    -min_tluplus ./path/best_settings.tlup
```

For multicorner-multimode designs, use the `set_tlu_plus_files` command to specify the TLUPlus files for each scenario. The `set_tlu_plus_files` command applies only to the current scenario. For information about setting the current scenario see “[Defining Scenarios](#)” on page 3-34. You can specify up to one hundred TLUPlus files per design.

If a TLUPlus file is not specified for each scenario, the tool issues an error similar to the following:

```
Error: tlu_plus files are not set in this scenario s1.
RC values will be 0.
```

To allow for temperature scaling, the TLUPlus files specified must contain the GLOBAL_TEMPERATURE, CRT1, and (optionally) CRT2 variables. The following example is an excerpt from a TLUPlus file:

```
TECHNOLOGY = 90nm.lib
GLOBAL_TEMPERATURE = 105.0
CONDUCTOR metal8 {THICKNESS= 0.8000
    CRT1=4.39e-3 CRT2=4.39e-7
...}
```

After specifying the TLUPlus files, you should validate them by running the `check_tlu_plus_files` command. An invalid TLUPlus setup can cause errors when you process your design. When you validate the TLUPlus files, the tool checks the consistency of conducting layer names and via layer names across the ITF file, Milkyway file, and mapping files. It also verifies that the minimum width and minimum spacing rules for metal layers are consistent between the ITF file and the Milkyway technology file, but it does not check minimum width and minimum spacing for via layers. IC Compiler uses the technology file as the reference for these checks. The consistency check compares the parameters in the TLUPlus files with the corresponding fields in the technology file.

Note:

The tool does not save the TLUPlus settings in the Milkyway design unless the design has a valid floorplan. For information about annotating a floorplan on the design, see “[Annotating the Floorplan Information](#)” on page 3-20. For information about creating a floorplan, see the *IC Compiler Design Planning User Guide*.

Table 3-2 lists the commands used for working with the TLUPlus file settings and their functions.

Table 3-2 Commands for Working With the TLUPlus File Settings

Task	Command	GUI
Define the TLUPlus file settings	<code>set_tlu_plus_files</code>	File > Set TLU+
Validate the TLUPlus file settings	<code>check_tlu_plus_files</code>	N/A
Report the TLUPlus file settings	<code>report_tlu_plus_files</code>	N/A

Specifying the Operating Conditions

The operating conditions of a design include the process, voltage, and temperature parameters under which the chip is intended to operate. The tool analyzes and optimizes the design under the conditions you specify.

The logic library can specify multiple sets of chip operating conditions (process, voltage, and temperature). To get a list of the operating conditions available in a particular library and to

view their characteristics, use either the `report_operating_conditions` or `report_lib-operating_condition` command.

To set the operating conditions for timing analysis and optimization, use the `set_operating_conditions` command. When you run this command, you can specify either a maximum operating condition only or both a minimum and a maximum operating condition (`-min` and `-max` options).

For multicorner-multimode designs, specify an operating condition for each scenario in the design. If an operating condition is not specified, MV-20 and MV-21 warning messages are issued as shown in the following example:

```
icc_shell> set_operating_conditions SLOW_95 -library max_v95_t125
icc_shell> create_scenario s1
Warning: Any existing scenario-specific constraints are discarded.
(MV-020)
icc_shell> report_timing
Warning: No operating condition was set in scenario s1 (MV-021)
```

Note:

Multicorner-multimode design libraries do not support the use of k-factor scaling. Therefore, the operating conditions that you specify for each scenario must match the nominal operating conditions of one of the libraries in the link library.

Selecting the Operating Condition Analysis Mode

The IC Compiler tool offers three analysis modes with respect to operating conditions:

- Single operating condition (default)

In the single operating condition mode, the tool uses a single set of delay parameters for the whole circuit, based on one set of process, temperature, and voltage conditions.

- Best-case and worst-case (`-analysis_type bc_wc` option)

In the best-case and worst-case mode, the tool simultaneously checks the circuit for the two extreme operating conditions: minimum and maximum. For setup checks, it uses maximum delays for all paths. For hold checks, it uses minimum delays for all paths. This mode lets you check both extremes in a single analysis run, thereby reducing overall runtime for a full analysis.

- On-chip variation (`-analysis_type on_chip_variation` option)

In the on-chip variation mode, the tool performs a conservative analysis that allows both minimum and maximum delays to apply to different paths at the same time. For a setup check, it uses maximum delays for the launch clock path and datapath and minimum delays for the capture clock path. For a hold check, it uses minimum delays for the launch clock path and datapath and maximum delays for the capture clock path. This mode models the effects of variation in operating conditions across the chip.

Preparing for Minimum and Maximum Timing Analysis

By default, when you perform simultaneous minimum and maximum timing analysis, IC Compiler uses the logic libraries specified by the `link_library` variable for both maximum and minimum timing information.

The tool also supports the following methods for determining the minimum and maximum timing information:

- Using two libraries
- Using derating factors
- Using advanced on-chip variation

Using Two Libraries

You specify separate minimum timing libraries by using the `set_min_library` command. The `set_min_library` command associates minimum timing libraries with the maximum timing libraries specified in the `link_library` variable. For example,

```
icc_shell> set_app_var link_library "* maxlib.db"
icc_shell> set_min_library maxlib.db -min_version minlib.db
```

Note:

The link path should contain only the maximum library, as shown in the previous example.

The `set_min_library` command is not scenario-specific. So, if you use this command to associate a minimum library to a specific maximum library, that relationship applies to all scenarios.

A minimum library can be associated with a single maximum library. Similarly, a maximum library can be associated with a single minimum library. As shown in the example in [Table 3-3](#), the minimum library `Fast_0yr.db` is associated with the maximum library, `Slow.db` of scenario s1. The minimum library `Fast_10yr.db` is associated with the maximum library, `SlowHV.db` of scenario s2.

Table 3-3 Supported Minimum-Maximum Library Configuration

Scenarios		
	s1	s2
Maximum library	Slow.db	SlowHV.db
Minimum library	Fast_0yr.db	Fast_10yr.db

The tool does not support associating a single minimum library with multiple maximum libraries or associating a single maximum library with multiple minimum libraries. If you use multiple `set_min_library` commands to associate a single minimum (or maximum) library with multiple maximum (or minimum) libraries, the tool uses the values specified in the last `set_min_library` command, overriding the values specified in the previous `set_min_library` command.

Consider the example shown in [Table 3-4](#), where the maximum library, Slow.db is associated with two minimum libraries, Fast_0yr.db and Fast_10yr.db, in scenarios s1 and s2 respectively. The tool does not support this type of association.

Table 3-4 Unsupported Multiple Minimum Library Configuration

Scenarios	
s1	s2
Maximum library	Slow.db
Minimum library	Fast_0yr.db

Example 3-5 Unsupported Multiple Minimum Library Configuration Script

```
create_scenario S1
set_min_library -min_version Fast_0yr.db Slow.db
...
create_scenario S2
set_min_library -min_version Fast_10yr.db Slow.db
...
```

In [Example 3-5](#), multiple `set_min_library` commands are used to associate a single maximum library with multiple minimum libraries. The tool overrides the value set by the first `set_min_library` command and issues the following warning message:

Warning: overriding result from previous `set_min_library` command on library 'Fast_0yr.db'.

For more information, see “Using Two Libraries for Min-Max Analysis” in the *Synopsys Timing Constraints and Optimization User Guide*.

To find out which libraries are defined as the maximum and minimum libraries, use the `list_libs` command. In the generated report, the uppercase letter “M” appears next to the maximum library, and the lowercase letter “m” appears next to the minimum library.

Using Derating Factors

If your timing libraries do not include both minimum and maximum timing data, you can perform simultaneous minimum and maximum timing analysis by specifying derating factors for your timing library.

Use the `set_timing_derate` command to specify the derating factors. For example, to specify that for the maximum operating condition, all early (shortest-path) delays should be decreased by 10 percent and all late (longest-path) delays should be increased by 20 percent, enter the following commands:

```
icc_shell> set_timing_derate -max -early 0.9  
icc_shell> set_timing_derate -max -late 1.2
```

By default, the `set_timing_derate` command applies the derating factors to the whole design. You can optionally specify a list of leaf-level cells or library cells, which restricts the scope of the command to only the listed cells.

In multi-scenario timing analysis, the command applies only to the scenario in which it is used, so you can set different derating factors for each scenario, except when you apply the command to library cells. By default, derating factors set on library cells affect the usage of those cells in all scenarios. To have the derating factors apply to only the current scenario, set the `timing_library_derate_is_scenario_specific` variable to `true`.

To report the derating factors, use the `report_timing_derate` command. To reset the derating factors to 1.0, use the `reset_timing_derate` command.

For more information, see “Setting Derating Factors” in the *Synopsys Timing Constraints and Optimization User Guide*.

Using Advanced On-Chip Variation

Advanced on-chip variation (AOCV) is an optional method of accuracy improvement that determines varying derating factors for different clock paths based on the path lengths. To use the feature, you create tables of derating values to be applied to clock paths in the design. For details, see “Advanced On-Chip Variation Analysis” in the *Synopsys Timing Constraints and Optimization User Guide*.

Setting Voltage and Temperature Scaling Between Libraries

The tool can use “scaling library groups” to implement voltage and temperature scaling. To get the cell delays at intermediate voltage and temperature conditions, the tool interpolates between the data in separate libraries that have been characterized at different nominal voltage and temperature values.

To use this feature, the libraries in the scaling group must contain CCS timing models or mixed CCS and NLDM timing models. The cell data must be consistent with respect to cell names, the number and names of pins, and the number and names of timing arcs.

Use the `define_scaling_lib_group` command to create a scaling library group and specify which libraries belong to the group. To apply a defined scaling library group to the whole design or to selected objects in the design, use the `set_scaling_lib_group` command. To create an intermediate operating condition, use the `create_operating_conditions` command.

For example,

```
icc_shell> define_scaling_lib_group -name group1 \
    { max_v70_t125.db max_v108_t125.db }
icc_shell> set_scaling_lib_group group1
icc_shell> set_operating_conditions SLOW_108 -library max_v108_t125
icc_shell> create_operating_conditions -name SLOW_95 \
    -library max_v70_t125 -process 1 -voltage 0.95 -temperature 125
icc_shell> set_operating_conditions SLOW_95 \
    -object_list { mid1/bot1 top_i }
```

This example creates a one-dimensional scaling group consisting of two libraries characterized at two extreme voltages, 0.70 and 1.25 volts. It applies this scaling group to the design and selects the 1.25-volt library as the default operating condition for the design. Then it creates a new operating condition at 0.95 volts and applies it to two objects in the design. The tool performs interpolation between the 0.70-volt and 1.25-volt libraries to obtain the timing characteristics at 0.95 volts and uses that information for timing analysis of the two objects.

To perform both voltage and temperature scaling at the same time, you would use four libraries in the scaling group rather than two, representing the four possible combinations of voltage and temperature extremes: high voltage and high temperature, high voltage and low temperature, low voltage and high temperature, and low voltage and low temperature.

To remove the scaling library group from a set of ports or cells, use the `remove_scaling_lib_group` command.

Note:

The `check_library` command does not check for consistency among the libraries in the scaling library group.

Setting Timing Constraints

At a minimum, the timing constraints must contain a clock definition for each clock signal, as well as input delay or output delay for each I/O port. This requirement ensures that all signal paths are constrained for timing.

To model the clock tree effects for placement before running clock tree synthesis, you should also define the uncertainty, latency, and transition constraints for each clock by using the `set_clock_uncertainty`, `set_clock_latency`, and `set_clock_transition` commands.

You can use a file of Synopsys Design Constraints (SDC) commands or use individual SDC commands. For details about the SDC commands, see the *Using the Synopsys Design Constraints Format Application Note*.

To read a timing constraints file, use the `read_sdc` command (or choose File > Import > Read SDC).

```
icc_shell> read_sdc -version 1.7 design_name.sdc
```

Important:

If the SDC file does not contain unit settings, they are derived from the main library (see [“Setting Up the Logic Libraries” on page 3-3](#)). If the SDC file does contain unit settings, they must be consistent with those in the main library.

The tool does not optimize paths that are not constrained for timing. Before proceeding, use the `check_timing` command to verify that all paths are constrained. If the `check_timing` command reports unconstrained paths, run the `report_timing_requirements` command to verify that the unconstrained paths are false paths (the `check_timing` command considers false paths unconstrained).

To remove the timing constraints, use one of the following commands:

- `remove_sdc`

This command removes the timing constraints set by SDC commands.

Note:

If you have run extraction on your design, you can prevent removal of the extracted RC network by using the `-keep_parasitics` option when you run the `remove_sdc` command. Keeping the RC network saves runtime because the tool does not need to redo the extraction.

- `remove_ideal_network -all`

This command removes `ideal_network` attributes, latencies, and transition times.

- `reset_design`

This command removes all attributes from the design, including timing constraints, optimization attributes, and floorplan information.

For multicorner-multimode designs, you must define timing constraints for each scenario. In general, when you set a timing constraint, it only applies to the current scenario. For information about setting the current scenario see [“Defining Scenarios” on page 3-34](#).

Selecting the Delay Calculation Method

By default, the tool uses the Elmore delay model for preroute delay calculation and the Arnoldi delay model for routed clock and postroute delay calculation. To change the delay calculation model, use the `set_delay_calculation_options` command.

For preroute delay calculation, the tool computes delays based on estimated parasitic data for the nets. You can choose either Elmore or asymptotic waveform evaluation (AWE) as the delay model. The AWE delay model provides better accuracy and better correlation with postroute delay calculation. If you choose the AWE delay model, you can optionally specify the effort level as low, medium (the default), or high. For example, to select the AWE delay model for preroute nets and set the effort to high, use the following command:

```
icc_shell> set_delay_calculation_options -preroute awe -awe_effort high
```

For routed clock delay calculation and for postroute delay calculation, you can choose either Arnoldi or Elmore as the delay model. The Arnoldi delay model provides better accuracy and better correlation with PrimeTime delay calculation, especially for smaller geometries and for highly resistive nets. If you choose the Arnoldi delay model, you can optionally specify the effort level as low, medium (the default), or high.

For example, to enable Arnoldi delay calculation for clock nets only, use the following command:

```
icc_shell> set_delay_calculation_options -routed_clock arnoldi
```

To enable Arnoldi delay calculation for postroute delay calculation, use the following command:

```
icc_shell> set_delay_calculation_options -postroute arnoldi
```

Note:

Using the Arnoldi delay model removes existing back-annotation stored in the database, unlike using the Elmore delay model. If you save a design with Arnoldi delay calculation, the delay information is not saved in the Milkyway database.

You can use the `compare_delay_calculation` command to compare Elmore and Arnoldi results of sample delay calculations in your design. If the results are similar between the two methods, use the faster Elmore method. Otherwise, if you need the additional accuracy, use the Arnoldi method.

By default, the tool does not include crosstalk delta delays in the delay calculations. To extract coupling capacitances and include crosstalk delta delays in the postroute delay calculations, enter the following command:

```
icc_shell> set_si_options -delta_delay true
```

Modifying the RC Coefficients for Preroute RC Estimation

The IC Compiler tool automatically performs preroute RC estimation when you run any placement or placement-based optimization command. When performing preroute RC estimation, the tool performs the following tasks:

- Determines the RC coefficients

The tool derives the RC coefficients from the TLUPlus models. For information about attaching the TLUPlus data to your Milkyway design library, see [“Setting Up the TLUPlus Files” on page 3-40](#).

The TLUPlus models contain layer-specific RC coefficients. Because layers are not assigned for preroute estimation, the tool calculates the RC coefficients by applying averaging techniques to the layer-specific RC coefficients.

- Estimates the routing topology

The tool uses virtual route technology to estimate the routing topology from the placement information. The virtual route does not contain layer assignments.

- Calculates the RC values

The tool uses virtual routing to estimate the preroute parasitic values, including the via resistance values.

- Performs delay estimation

The tool uses the Elmore delay model to estimate the delay values, based on the estimated routing topology and the estimated RC values.

- Saves the timing data and attributes in the Milkyway design library

The tool saves total capacitance as a design attribute.

You can explicitly perform preroute RC estimation by running the `extract_rc` command on a placed design. By default, the `extract_rc` command performs RC extraction on any unrouted nets in your design and performs RC extraction on the routed nets in your design. If your design contains a mix of routed and unrouted nets, and you do not want to run RC extraction on the routed nets, use the `extract_rc -estimate` command, which performs only RC estimation.

When the tool runs preroute RC estimation, it displays the derived coefficients on your screen. You can modify these coefficients by using the following methods:

- Defining net-based layer constraints

At small geometries, the RC coefficients, especially resistance coefficients, can vary significantly between layers. In this case, the coefficients derived by averaging the coefficients for all layers might not provide sufficient correlation with the postroute values. You can restrict the set of routing layers used for a specific net by defining net-based layer constraints. For more information, see “[Defining Net-Based Layer Constraints](#)” on [page 3-52](#).

- Introducing pessimism

If, after analyzing the estimation results calculated from the derived RC coefficients and via resistance values, you feel that the derived RC coefficients are too optimistic, you can consider introducing pessimism to these values. Using pessimistic values means that you provide larger values, so that for a given net, the tool computes a larger delay. This, in turn, drives the optimization of your design such that extra buffering or more powerful driving cells are introduced. For more information, see “[Introducing Pessimism](#)” on [page 3-52](#).

- Defining the pattern density outside the block

In StarRC, the `DENSITY_OUTSIDE_BLOCK` command specifies the pattern density outside the block, which affects the thickness variation and parasitic RC values. For correct correlation with StarRC, specify the same pattern density value in the IC Compiler tool by using the `-density_outside_block` option of the `set_delay_estimation_options` or `set_extraction_options` command. For more information about specifying the pattern density outside the block, see the description of the `DENSITY_OUTSIDE_BLOCK` command in the *StarRC User Guide and Command Reference*.

To report the delay estimation coefficients, use the `report_delay_estimation_options` command.

Defining Net-Based Layer Constraints

In addition to restricting the layers to which a specific net can be assigned during routing, net-based layer constraints improve RC correlation by reducing the set of values used by the averaging techniques that determine the RC coefficients used for preroute estimation.

To set net-based layer constraints, use the `set_net_routing_layer_constraints` command.

```
icc_shell> set_net_routing_layer_constraints {netA} \
-min_layer_name metal1 -max_layer_name metal3
```

The routing layer names you specify must exist in both the physical library and the design, and the maximum routing layer must be the same or on top of the minimum routing layer.

If the net you specify already has routing layer constraints set, a warning appears and the constraints specified by this invocation overwrite the existing routing layer constraints.

To report and remove routing layer constraints, use the `report_net_routing_layer_constraints` and `remove_net_routing_layer_constraints` commands.

Introducing Pessimism

You can introduce pessimism either by scaling the derived RC coefficients and via resistance values or by manually specifying the RC coefficients and via resistance values. Avoid more than doubling the default numbers, because the gate sizes and runtime increase. The basic rule is to increase values by 10 percent (multiply by 1.1). Usually, change the number by small amounts, such as increments of 0.1.

Scaling the RC Coefficients

To scale the RC coefficients,

1. Determine the scaling factor you want to apply.
2. Use the following `set_delay_estimation_options` command options to set the scaling factors:

```
-max_unit_horizontal_capacitance_scaling_factor
-max_unit_vertical_capacitance_scaling_factor
-max_unit_horizontal_resistance_scaling_factor
-max_unit_vertical_resistance_scaling_factor
-min_unit_horizontal_capacitance_scaling_factor
-min_unit_vertical_capacitance_scaling_factor
-min_unit_horizontal_resistance_scaling_factor
-min_unit_vertical_resistance_scaling_factor
```

The tool produces new RC coefficients by multiplying the derived RC coefficients by the scaling factors you specify. During RC estimation, the tool uses the scaled coefficients.

Scaling the Via Resistance Values

To scale the via resistance values,

1. Determine the scaling factor you want to apply.
2. Use the following `set_delay_estimation_options` command options to set the scaling factors:

```
-max_via_resistance_scaling_factor  
-min_via_resistance_scaling_factor
```

The tool produces new resistance values by multiplying the derived values by the scaling factors you specify. During delay estimation, the tool uses the scaled values.

Manually Specifying RC Coefficients

To specify the RC coefficients,

1. Run RC estimation to derive the base RC coefficients.
2. Determine the new, pessimistic values you want to apply.
3. Use the following `set_delay_estimation_options` command options to specify the RC coefficients:

```
-max_unit_horizontal_capacitance  
-max_unit_vertical_capacitance  
-max_unit_horizontal_resistance  
-max_unit_vertical_resistance  
-min_unit_horizontal_capacitance  
-min_unit_vertical_capacitance  
-min_unit_horizontal_resistance  
-min_unit_vertical_resistance
```

Note:

At the minimum, you must specify all four `-max_*` options; otherwise, the tool uses the derived RC coefficients.

The values you specify override the derived RC coefficients. During RC estimation, IC Compiler uses the specified coefficients.

If you specify both scaling factors and user-defined RC coefficients, the resulting RC coefficients are the result of multiplying your RC coefficients by your scaling factors.

For multicorner-multimode designs, use the `set_delay_estimation_options` command to specify scaling factors and user-defined RC coefficients for each scenario. The `set_delay_estimation_options` command applies only to the current scenario. For information about setting the current scenario see “[Defining Scenarios](#)” on page 3-34.

Manually Specifying Via Resistance Values

To specify the via resistance values,

1. Run RC estimation to derive the base resistance values.
2. Determine the new, pessimistic values you want to apply.
3. Use the following `set_delay_estimation_options` command options to specify the resistance values:

```
-max_via_resistance  
-min_via_resistance
```

The values you specify override the derived via resistance values. During delay estimation, the tool uses the specified values.

Note:

If you specify both scaling factors and user-defined via resistance values, the resulting via resistance values are the result of multiplying your values by your scaling factors.

Back-Annotating Delay or Parasitic Data

If you have existing delay or parasitic data, you can back-annotate the design with this data instead of using the IC Compiler extraction capability to calculate the data.

To back-annotate the design with delay information provided in a Standard Delay Format (SDF) file, use the `read_sdf` command (or choose File > Import > Read SDF). To back-annotate the design with parasitic capacitance and resistance information, use the `read_parasitics` command. The tool accepts parasitic data in Synopsys Binary Parasitic Format (SBPF) or Standard Parasitic Exchange Format (SPEF).

Note:

If you read the parasitic data in SPEF format, the tool uses this data for analysis only. If you read the data in SBPF format, the tool can use the data for both analysis and postroute optimization.

To remove annotated data from your design, use the `remove_annotations` command.

Linking Designs

When the IC Compiler tool performs timing analysis, each cell instance in the design must be linked to a cell in the link libraries, which provides its timing information. For information about defining the link libraries, see “[Setting Up the Logic Libraries](#)” on page 3-3. In general, IC Compiler automatically links the design before performing timing analysis. If you want to explicitly link your design, use the `link -force` command.

Linking Designs Without Multicorner-Multimode Scenarios

If your design does not contain multicorner-multimode scenarios, the tool links each cell instance to the first reference cell it locates in the link libraries. If it locates additional reference cells with the same name, it generates a warning message identifying the ignored, duplicate reference cells. If the tool does not find the reference cell, a warning appears advising that the reference cell cannot be resolved.

Linking Designs With Multicorner-Multimode Scenarios

If your design contains multicorner-multimode scenarios, the tool groups the link libraries into sets by using their nominal process, voltage, and temperature (PVT) values. Libraries with the same nominal PVT values are grouped into the same set. This grouping scheme allows you to use logic libraries that do not have operating condition definitions. It also provides the flexibility of having multiple library files (for example, one for standard cells, another for macros, and so forth).

When you use multiple libraries,

- If library cells with the same name are not functionally identical or do not have identical sets of library pins with the same name and order, a warning is issued, stating that the libraries are inconsistent.
- If library cells with the same name have the same nominal PVT, a warning is issued, stating that the libraries are ambiguous. The warning also states which libraries are being used and which are being ignored.

When you set the operating conditions for a scenario, the tool uses the PVT values of the maximum operating condition to select the link library set associated with that scenario. For each scenario, the tool links the cell instances in the design with the first matching reference cell in its associated link library set. Note that using the `-library` option with the `set_operating_conditions` command merely helps the tool identify the PVT for the operating conditions; it does not specify the library to link.

Note:

Because the grouping is done by nominal PVT values, you must ensure that these values are distinct for the maximum libraries associated with each corner; otherwise, they are grouped into the same set. Because IC Compiler links cell instances to the first matching library cell in a set, the cells are not linked correctly, which results in incorrect timing values.

For example, assume that you have two sets of libraries, one for each corner, as shown in [Table 3-5](#). Each corner has three libraries: one for combinational cells, one for sequential cells, and one for macros.

Table 3-5 Link Libraries With Distinct PVT Values

Link library (in order)	Nominal PVT	Operating conditions in library (PVT)
Combo_cells_slow.db	1/0.85/130	WORST (1/0.85/130)
Sequentials_fast.db	1/1.30/100	None
Macros_fast.db	1/1.30/100	None
Macros_slow.db	1/0.85/130	None
Combo_cells_fast.db	1/1.30/100	BEST (1/1.3/100)
Sequentials_slow.db	1/0.85/130	None

In this case, the libraries are grouped as

- Combo_cells_slow, Macros_slow, and Sequentials_slow with PVT values of 1, 0.85, and 130, respectively.
- Combo_cells_fast, Macros_fast, and Sequentials_fast with PVT values of 1, 1.30, and 100, respectively.

To create a scenario named s1 with the cell instances linked to the Combo_cells_slow, Macros_slow, and Sequentials_slow libraries, enter the following command:

```
icc_shell> create_scenario s1
icc_shell> set_operating_conditions -max WORST -library Combo_cells_slow
```

In the following example, the link library contains four libraries, one for each corner, with each library containing the same library cells; however, the nominal PVT values are not distinct between these libraries, as shown in [Table 3-6](#).

Table 3-6 Link Libraries That Do Not Have Distinct PVT Values

Link library (in order)	Nominal PVT	Operating conditions in library (PVT)
Ftyp.db	1/1.30/100	WORST (1/1.30/100)
Typ.db	1/0.85/100	WORST (1/0.85/100)

Table 3-6 Link Libraries That Do Not Have Distinct PVT Values (Continued)

Link library (in order)	Nominal PVT	Operating conditions in library (PVT)
TypHV.db	1/1.30/100	WORST (1/1.30/100)
Holdtyp.db	1/0.85/100	BEST (1/0.85/100)

In this case, the libraries are grouped as

- Ftyp and TypHV with PVT values of 1, 1.30, and 100, respectively.
- Typ and Holdtyp with PVT values of 1, 0.85, and 100, respectively.

When you run the following commands, the cell instances in scenario s2 are not linked to the library cells in the TypHV.db library, as intended. Instead, they are linked to the library cells in the Ftyp.db library, which is the first logic library in its link set.

```
icc_shell> create_scenario s1
icc_shell> set_operating_conditions WORST -library Typ.db
icc_shell> create_scenario s2
icc_shell> set_operating_conditions WORST -library TypHV.db
icc_shell> create_scenario s3
icc_shell> set_operating_conditions WORST -library Ftyp.db
icc_shell> create_scenario s4
icc_shell> set_operating_conditions \
    -max WORST -max_library Typ.db -min BEST -min_library HoldTyp.db
```

Resolving References With Operating Condition Mismatches

Before doing timing analysis, you can check whether your design contains any cells for which the link libraries do not have cells characterized with the correct operating conditions by running the `check_mv_design -opcond_mismatches` command.

By default, when the tool cannot find an exact match for the reference cell name and operating conditions, it tries to match the reference cell with the same name and the closest operating conditions using the following process:

1. Find the set of library cells with the same name as the reference cell.

Note:

If the cell instance is connected to a power net, reference cells whose rail voltages do not match the explicit power connection rails are not included in this set.

2. Identify the library cells in this set that match each of the following criteria, in order.
 - a. The PVT values of the library cell match the PVT values of the design.
 - b. The process, temperature, and voltage values from one of the rails of the library cell match the PVT values of the design.
 - c. The temperature and voltage values of the library cell match the temperature and voltage values of the design.
 - d. The process and voltage values of the library cell match the process and voltage values of the design.
 - e. The voltage value of the library cell matches the voltage value of the design.
 - f. The voltage value from one of the rails of the library cell matches the voltage value of the design.
 - g. The process and temperature values of the library cell match the process and temperature values of the design.
 - h. None of the process, voltage, and temperature values of the library cell match the process, voltage, and temperature values of the design.

The process stops at the first criteria where either a single library cell matches or multiple library cells match.

If a single library cell matches, the tool links that library cell to the cell instance, uses its PVT values as the operating conditions of the cell instance, and issues the warning following message:

```
Warning: cell top/macro1(mac1) was constrained to operate under the
operating condition (voltage = 1.110000V, process = 1.000000,
temperature = 125.000000), but the library cell mac1 matching this
characterization cannot be found in the link libraries. Instead,
library cell mac1 from mac1_1.00v_125c.db:mac1_1.00v_125c (voltage =
1.000000V, process = 1.000000, temperature = 125.000000) has been
used. (LIBSETUP-001)
```

If multiple library cells match, an error occurs and the cell instance remains unresolved.

If none of the criteria result in a single matching library cell, an error occurs and the cell instance remains unresolved.

For a macro-cell, pad-cell, or switch-cell instance, if you do not want to use this default resolution, use the `set_opcond_inference` command to control the degree to which the inferred operating condition can deviate from the operating condition of the design. A higher degree of deviation results in a lower likelihood of unresolved references; however, it can result in reduced timing accuracy.

You can reduce the degree of deviation by

- Requiring the process and temperature values of the library cell to match the process and temperature values of the design
Set the `-match_process_temperature` option to `true` to enable this requirement.
- Requiring a single name-based match within the link libraries
Set the `-level` option to `unique_resolved` to enable this requirement. Note that if the cell instance is connected to a power net, a rail voltage of the library cell must match the explicit power connection rail voltage.
- Requiring an exact match of both name and operating conditions
Set the `-level` option to `exact` to enable this requirement.

To increase the degree of deviation, set the `-level` option to `closest_unresolved`. With this setting, the library cell matching process is the same as the default process (`-level closest_resolved`); however, when multiple library cells match a criteria, the first matching library cell is selected.

By default, the settings of the `set_opcond_inference` command apply to all macro-cell, pad-cell, and switch-cell instances. You can restrict the types of cells to which the settings apply by using the `-applies_to` option. You can apply the settings to specific cell instances by using the `-object_list` option.

Note:

The `set_opcond_inference` command applies only to macro-cell, pad-cell, and switch-cell instances. It does not apply to any other type of leaf-cell instance in your design.

Saving a Design

You can save a design for use with Milkyway design tools, for use with other tools, and for manufacturing. If you are planning to work further with the design in the IC Compiler tool or another design tool based on Milkyway, save the design as explained in “[Saving the Design in Milkyway Format](#)” on page 3-60. If you are going to use a tool that does not use Milkyway, export the design to ASCII format (which includes Verilog, DEF, SDC, and parasitic files), as explained in “[Saving the Design in ASCII Format](#)” on page 3-63. If you have completed the design and are ready for manufacturing, save it in GDSII or Oasis format, as explained in “[Writing GDSII and OASIS Layout Data Files](#)” on page 3-64.

Saving the Design in Milkyway Format

To save the design in Milkyway format, use the `save_mw_cel` command (or click the  button on the File toolbar or choose File > Save Design).

When you save a design, the tool automatically updates the Milkyway database. If your design does not have unit settings, the tool applies the default settings from the main library and saves them with the design. For more information about the main library and its default settings, see “[Setting Up the Logic Libraries](#)” on page 3-3.

Saving the Design From the Command Line

When you enter the `save_mw_cel` command on the shell command line, if you do not specify a design, the tool saves the current Milkyway design. To save another design, specify the design name on the command line.

```
icc_shell> save_mw_cel design_name
```

By default, the top-level design is saved with the same name and CEL version.

To increment the CEL version, use the `-increase_version` option.

```
icc_shell> save_mw_cel design_name -increase_version
```

To save a copy of the design with a different name, use the `-as` option.

```
icc_shell> save_mw_cel design_name -as new_name
```

To save the top-level design together with its subdesigns and attached files in a single CEL view, use the `-hierarchy` option.

```
icc_shell> save_mw_cel design_name -hierarchy
```

By default, the `save_mw_cel` command saves all scenarios with a multicorner-multimode design. To save specific scenarios with the design, use the `-scenarios` option.

```
icc_shell> save_mw_cel design_name -scenarios scenario_names
```

Saving the Design in the GUI

When you choose File > Save Design in the GUI, the Save Design dialog box appears.

- To save the top-level design for all modified designs with the same name and CEL version, click Save All.
- To save the designs using other options, select “Show advanced options,” which displays additional options and additional columns in the list of designs.
 - (Optional) To display only the names of designs with unsaved changes, select the “All modified cells” option.
The “All open cells” option is selected by default.
 - (Optional) To specify a list of the scenarios with constraints that you need to save with the design, type the scenario names in the Scenarios box.
Alternatively, you can click the browse button and select the scenarios in the Scenarios Chooser dialog box.
 - Set options to save or discard design changes as needed.

You can select which designs you want to save and which methods to use to save them by setting options in the design list.

- To save the design, select the Save option.

By default, this option is selected for designs that have unsaved changes and deselected for designs that do not have unsaved changes. If you do not want to save the changes for a design, deselect its Save option.

- To increment the CEL version, select the “New version” option.
- To save a modified top-level design together with all its subdesigns and attached files in the same CEL version, select the Save Hierarchy option.

The subdesigns must be in the same design library as the top-level design. Subdesigns that are instantiated from the reference libraries are not included.

This option is not available when the “New version” or Save As option is selected.

- To save a copy of a design, select the Save As option and type a new design name in the Save As Name box.

Do not include a period (.) in the name unless you want to include the CEL view name. In Milkyway, the period is a special character used to separate design names from view names. A period followed by an invalid view name causes an error.

- To close a design after saving it, select the Close option.

If you need to close a design and discard the changes, select the Close option and deselect the Save option.

You can select or deselect options for all designs by using the buttons below the list.

- To select the Save options for all designs, click the Select Save All button.
- To deselect the Save options for all designs, click the Discard All Changes button.
- To select the Save Hierarchy options for all designs, click the Select Hierarchy All button.
- To deselect the Save Hierarchy options for all designs, click the Clear Hierarchy All button.
- Click OK or Apply.

Note:

When you use the GUI to close designs, close the design library, or exit the tool, IC Compiler opens a dialog box similar to the Save Design dialog box and gives you the option of automatically saving the open designs in Milkyway format.

Saving the Design Settings in the Milkyway Design Library

By default, the design settings, which are controlled by setting variables such as `search_path` and `link_library`, are not stored in the Milkyway design library. Rather than save these settings in a separate script file, you can use the `save_design_settings` command to save them in the design library, which helps to ensure that the correct settings are used in subsequent sessions.

To save the design settings in the Milkyway design library that is currently open, use the `save_design_settings -library` command.

```
icc_shell> save_design_settings -library
```

By default, the command saves all the variable settings that have been changed from their defaults. It saves them as a sequence of `set_app_var` commands. However, if you use the `-input` option, it saves the sequence of commands contained in the specified text file into

the design library, without considering whether these commands change default variable values. The file can contain any commands, not just `set` or `set_app_var` commands.

To report the commands that have been saved into the design library by the `save_design_settings` command, use the `write_design_settings -library` command. You can optionally write this command sequence into a text file by using the `-output` option.

```
icc_shell> write_design_settings -library -output my_settings.tcl
```

The `restore_design_settings -library` command restores the design settings previously stored into the design library with the `save_design_settings` command. It reads the previously saved commands from the library and runs them.

```
icc_shell> restore_design_settings -library
```

In each of these commands, you must use the `-library` option so that the design settings are saved to or restored from the Milkyway design library that is currently open.

Saving the Design in ASCII Format

To save the design in ASCII format,

1. Update the power and ground connections by using the `derive_pg_connection` command. This is a required step when writing a Verilog netlist. For more information, see “[Creating Logical Power and Ground Connections](#)” on page 3-26.
2. Ensure that the object names in the design are Verilog-compliant by using the `change_names` command. This is a required step when writing a Verilog netlist.

```
icc_shell> change_names -hierarchy -rules verilog
```

3. Write the Verilog data for the design by using the `write_verilog` command.

```
icc_shell> write_verilog design_name.v
```

By default, the generated Verilog netlist does not include the power and ground nets. To include the power and ground nets, use the `-pg` and `-output_net_name_for_tie` options when you run the `write_verilog` command.

Note:

Before you run the `write_verilog` command, use the `check_database` command to validate the infrastructure integrity of the Milkyway design database. For more information, see “[Validating the Design](#)” on page 3-25.

4. Write the floorplan information to a DEF file by using the `write_def` command (or by choosing File > Export > Write DEF).

```
icc_shell> write_def -output design_name.def
```

5. Write the design constraints to SDC files by using the `write_sdc` command.

```
icc_shell> write_sdc design_name.sdc
```

If the SDC file does not have unit settings, the tool uses the default settings from the main library and saves them when you save the SDC file. For more information about the main library and its default settings, see “[Setting Up the Logic Libraries](#)” on page 3-3.

6. Write the RC extraction data to the parasitic netlist by using the `write_parasitics` command (or by choosing File > Export > Write Parasitics).

The `write_parasitics` command can write parasitic netlists in the SPEF and SBPF formats. Specify the format by using the `-format` option (or the “Net format in file” box in the GUI). You can also compress the output in gzip format by using the `-compress` option (or by selecting “Write in gzip format” in the GUI).

```
icc_shell> write_parasitics -format SPEF -output design_name.spef
```

Writing GDSII and OASIS Layout Data Files

After you have completely finished and checked the chip design, you can write out the design in GDSII or OASIS format for delivery to the fabrication facility. You set the options for layout file generation by using the `set_write_stream_options` command and generate the layout data file with the `write_stream` command.

The `set_write_stream_options` command lets you set many file generation options such as naming conventions, name mapping, hierarchical depth, via flattening, and file compression. You can use the command multiple times to set individual options incrementally.

After you set the options, you use the `write_stream` command to write out the design in GDSII or OASIS format. This is the syntax of the command:

```
write_stream
[-lib_name lib_name]
[-format gds | oasis]
[-cells list_of_cells]
[-cell_file cell_name_file]
stream_file_name
```

You must specify at least the output file name. The default format is GDSII, so to write the design in OASIS format, you must use the `-format oasis` option. For example,

```
icc_shell> write_stream -format oasis my_design.oasis
```

By default, the command writes out all cells in the currently opened library. To restrict the output to a specified library or specified cells, use the `-lib_name`, `-cells` or `-cell_file` option.

You can also read GDSII and OASIS layout data for library cell preparation tasks. This is the syntax of the `read_stream` command:

```
read_stream
  [-lib_name lib_name]
  [-format gds | oasis]
  stream_file_name
```

The behavior of this command is affected by the option settings made with the `set_read_stream_options` command. You can report these option settings with the `report_read_stream_options` command.

Closing Designs

You can close designs at any time during a session. If you have unsaved changes in one or more designs, you can save or discard the changes before closing the designs.

To close Milkyway designs, use the `close_mw_cel` command (or choose File > Close Design).

```
icc_shell> close_mw_cel list_of_designs
```

By default, when you close Milkyway designs by using the `close_mw_cel` command, IC Compiler does not save the designs. To save a new CEL version for each specified design, use the `-save` option with the `close_mw_cel` command.

When you close Milkyway designs by choosing File > Close Design in the GUI and there are unsaved changes in any open designs, the Close Design dialog box appears, which enables you to save the changes to the modified designs. By default, you can either save or discard all changes. If you select “Show advanced options,” you can

- Specify a list of scenarios with constraints that you need to save with the designs.
- Select from the following options for each modified design,
 - Save a new CEL version of the design
 - Save a copy of a design with a new name
 - Save the design hierarchy (a modified top-level design together with all its subdesigns and attached files) into the same CEL version

Note:

If there are no open designs with unsaved changes, the current Milkyway design is closed without displaying the Close Design dialog box.

For more information about using the advanced options to save or discard changes, see “[Saving the Design in the GUI](#)” on page 3-61.

Closing a Milkyway Design Library

To close a Milkyway design library, use the `close_mw_lib` command (or choose File > Close Library).

By default, when you close a Milkyway design library by using the `close_mw_lib` command, the tool does not save the open designs. To save new CEL versions for all open designs, use the `-save` option with the `close_mw_lib` command.

When you close a Milkyway design library by choosing File > Close Library in the GUI and there are unsaved changes in any open designs, the Close Library dialog box appears, which enables you to save the changes to the modified designs. By default, you can either save or discard all changes. If you select “Show advanced options,” you can

- Specify a list of scenarios with constraints that you need to save with the designs.
- Select from the following options for each modified design,
 - Save a new CEL version of the design
 - Save a copy of a design with a new name
 - Save the design hierarchy (a modified top-level design together with all its subdesigns and attached files) into the same CEL version
 - Close the design after saving or discarding changes

Note:

If there are no open designs with unsaved changes, the Milkyway design library is closed without displaying the Close Library dialog box.

For more information about using the advanced options to save or discard changes, see [“Saving the Design in the GUI” on page 3-61](#).

Archiving Designs

The `archive_design` command copies a specified design or list of designs into a new library. The new archive library is a smaller, self-contained set of files for the designs of interest, without the unwanted design files contained in the original source library. The copied design includes the Milkyway physical design files, associated logic libraries, TLUPlus files, and a `design_setup.tcl` file that sets the variables associated with the design.

To produce an archive copy of a design, first close the Milkyway library containing the design. Then run the `archive_design` command, as in the following example:

```
icc_shell> archive_design -source ~/my_designs/ORCA/design \
    -design "icc_top_place icc_top_route" -archive my_archive_dir
```

You must specify at least the source Milkyway design library (`-source`), the names of the designs to archive (`-design`), and the archive directory (`-archive`).

By default, the command copies all views in the full hierarchy of the design (CEL and FRAM views), copies only the reference libraries containing cells actually used in the designs, and within each library, copies only the cells actually used in the designs or their hierarchy.

You can override the default behavior by using the `archive_design` command options. For example, the following command copies all cells contained in the reference libraries, not just those used in the archived designs, and it also copies a specified reference library that is not currently being used by the archived designs:

```
icc_shell> archive_design -source ~/my_designs/ORCA/design \
    -design "icc_top_place icc_top_route" -archive my_archive_dir \
    -complete -include_ref_lib "/work/ref_libs/macro_cell.lib"
```

The `archive_design` command writes a script file called `design_setup.tcl` to the archive directory. This script sets the application variables for the session similar to the script file generated by the `write_app_var` command. The `search_path`, `target_library`, and `link_library` variables are automatically modified in the script to work with the new archive location.

4

Placement and Optimization

This chapter describes the IC Compiler placement and optimization capabilities.

This chapter contains the following sections:

- [Preparing for Placement and Optimization](#)
- [Performing Placement and Optimization](#)
- [Analyzing Placement and Optimization Results](#)
- [Fine-Tuning the Placement and Optimization Results](#)

Preparing for Placement and Optimization

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

- [Defining Placement Blockages](#)
- [Setting the Congestion Options](#)
- [Controlling Placement Density](#)
- [Defining Placement Bounds](#)
- [Preventing Cell Placement in the Default Voltage Area](#)
- [Defining Spacing Requirements](#)
- [Defining Target Library Subsets](#)
- [Defining the Buffer Strategy for Optimization](#)
- [Setting the Preferred Buffers for Hold Fixing](#)
- [Enabling Tie Cell Insertion](#)
- [Preserving dont_touch Nets During Optimization](#)
- [Limiting Fanout During Optimization](#)
- [Setting Placement and Optimization Attributes](#)
- [Preparing for High-Fanout Net Synthesis](#)
- [Specifying Layer Optimization Settings](#)
- [Preparing for Clock Tree Synthesis During Placement](#)
- [Inserting Port Protection Diodes](#)
- [Setting Up Multithreading](#)

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 placed 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.

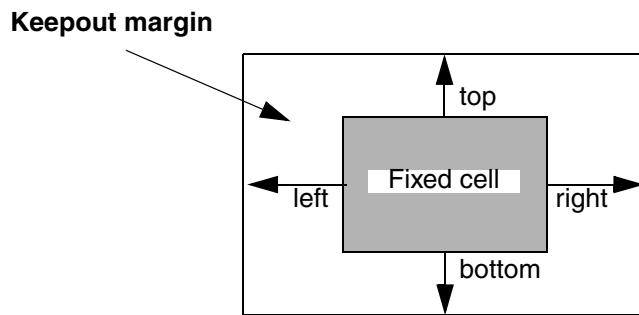
If you define both hard and soft placement blockages in your design, the hard placement blockages take priority over the soft placement blockages in places where they overlap.

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 4-1](#)) around the boundary of fixed macros in your design in which no other cells are placed.

Figure 4-1 Placement Keepout Margin



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.

The width of the keepout margin on each side of a fixed macro can be the same or different, depending on how you define the keepout margin.

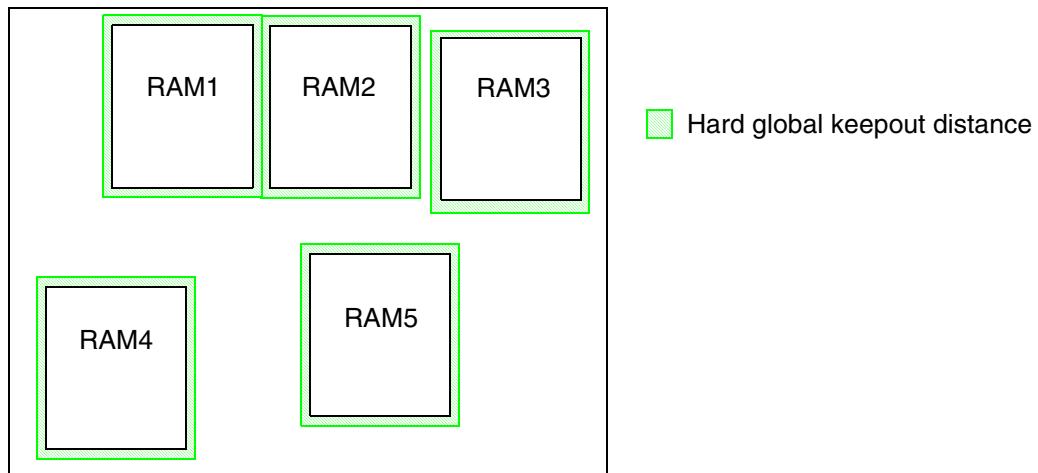
Defining Global Keepout Margins

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
```

[Figure 4-2](#) shows an example of using hard global keepout margins in a design containing five RAM cells. The hard keepout margins are applied to all macros.

Figure 4-2 Global Keepout Distances

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.

Instead of specifying the keepout margins explicitly, you can have them derived automatically by using `-tracks_per_macro_pin value`. In that case, the keepout margin is

calculated from the track width, the number of macro pins, and the specified track-to-pin ratio, which is typically set to a value near 0.5. A larger value results in larger keepout margins. The derived keepout margin is always hard; the `-type` setting is ignored. For derived margins, the `-all_macros`, `-macro_masters`, and `-macro_instances` options are not allowed. The derived margins are subject the minimum and maximum values specified by the `-min_padding_per_macro` and `-max_padding_per_macro` options. Here is an example:

```
icc_shell> set_keepout_margin -tracks_per_macro_pin 0.6 \
    -min_padding_per_macro 0.1 -max_padding_per_macro 0.2
```

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

An area-based placement blockage is a rectangular region in which cells cannot be placed or in which the types or number of cells is limited. To define an area-based placement blockage, use the `create_placement_blockage` command or choose Floorplan > Create Placement Blockage in the GUI. At a minimum, you must specify the coordinates of the bounding box for the placement blockage. You can optionally assign a name to a placement blockage by using the `-name` option. You can then reference that blockage by name to query or remove the placement blockage.

By default, the `create_placement_blockage` command creates a hard placement blockage. To create another type of placement blockage, use the `-type` option to specify the blockage type. A single `create_placement_blockage` command can create just one type of placement blockage.

The IC Compiler tool supports the following types of area-based placement blockages:

- Hard

A hard blockage prevents the placement of standard cells and hard macros within the specified area during coarse placement, optimization, and legalization. For information about defining a hard placement blockage, see “[Defining a Hard Placement Blockage](#)” on page 4-7.

- Hard macro

A hard macro blockage prevents the placement of hard macros within the specified area during coarse placement, optimization, and legalization. For information about defining a hard macro placement blockage, see “[Defining a Hard Macro Blockage](#)” on page 4-7.

- Soft

A soft blockage prevents the placement of standard cells and hard macros within the specified area during coarse placement, but allows optimization and legalization to place cells within the specified area. For information about defining a soft placement blockage, see “[Defining a Soft Placement Blockage](#)” on page 4-7.

- Partial

A partial blockage limits the cell density in the specified area during coarse placement, but has no effect during optimization and legalization. For information about defining a partial placement blockage, see “[Defining a Partial Placement Blockage](#)” on page 4-8.

- Pin

A pin blockage prevents the global router from routing in the specified area, and the pin placer from assigning pins to the area. For information about defining a pin placement blockage, see “[Defining a Pin Blockage](#)” on page 4-10.

Defining a Hard Placement Blockage

To define a hard placement blockage, specify the bounding box and optionally a name for the placement blockage.

For example, to create a hard placement blockage in the area enclosed by a rectangle with corners at (10, 20) and (100, 200), use the following command:

```
icc_shell> create_placement_blockage -bbox {10 20 100 200}
```

Hard placement blockages can also be defined in the DEF as shown in [Example 4-1](#).

Example 4-1 Placement Blockages in DEF

```
BLOCKAGES 2 ;
- PLACEMENT
  RECT ( 0 327600 ) ( 652740 327660 ) ;
- PLACEMENT
  RECT ( 0 327600 ) ( 652740 327660 ) ;
END BLOCKAGES
1
```

Defining a Hard Macro Blockage

A hard macro blockage prevents the placement of hard macros within the specified area, but not standard cells. To create a hard macro blockage, use the `-no_hard_macro` option.

To define a hard macro blockage, specify the bounding box, type (`-no_hard_macro` option), and optionally the name for the placement blockage.

For example, to define a hard macro blockage enclosed by a rectangle with corners at (120, 75) and (230, 200), use the following command:

```
icc_shell> create_placement_blockage -bbox {120 75 230 200} -no_hard_macro
```

Defining a Soft Placement Blockage

To define a soft blockage, specify the boundary, type (`-type soft` option), and optionally the name for the placement blockage.

For example, to define a soft blockage enclosed by a rectangle with corners at (120, 75) and (230, 200), use the following command:

```
icc_shell> create_placement_blockage -bbox {120 75 230 200} -type soft
```

Defining a Partial Placement Blockage

To define a partial blockage, specify the boundary, type (`-type partial` option), blockage percentage (`-blocked_percentage` option), and optionally the name for the placement blockage.

For example, to define a partial blockage with a maximum allowed cell density of 60 percent (a blocked percentage of 40), enclosed by the rectangle with corners at (10, 20) and (100, 200), use the following command:

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

Note:

To allow unlimited usage of a partial blockage area, specify a blockage percentage of zero (`-blocked_percentage 0` option).

In addition to controlling the maximum cell density of partial blockage, you can also control the types of cells that are placed within the partial blockage area. Note that the following options are mutually exclusive, and must be specified along with the `-type partial` option.

- Buffer-only

A buffer-only blockage is a special type of partial blockage that allows only buffers and inverters to be placed within the blockage. To create a buffer-only blockage, use the `-type partial`, `-blocked_percentage`, and `-buffer_only` options.

For example, to define a partial blockage that allows only the placement of buffers and inverters, with a cell density of 30 percent, use the following commands:

```
icc_shell> create_placement_blockage -bbox {10 20 100 200} \
    -type partial -blocked_percentage 70 -buffer_only
```

- No-register

A no-register blockage is a special type of partial blockage that prevents registers from being placed within the blockage. To create a no-register blockage, use the `-type partial`, `-blocked_percentage`, and `-no_register` options.

For example, to define a partial blockage that excludes registers, but allows a cell density of 50 percent for other cells, use the following commands:

```
icc_shell> create_placement_blockage -bbox {10 20 100 200} \
    -type partial -blocked_percentage 50 -no_register
```

- No-relative-placement

A no-relative-placement blockage is a special type of partial blockage that prevents relative placement groups from being placed within the blockage. To create a

no-relative-placement blockage, use the `-type partial`, `-blocked_percentage`, and `-no_rp_group` options.

For example, to define a partial blockage that excludes relative placement groups, but allows a cell density of 100 percent for all other cells, use the following commands:

```
icc_shell> create_placement_blockage -bbox {10 20 100 200} \
-type partial -blocked_percentage 0 -no_rp_group
```

- Category

A category blockage is a special type of partial blockage that controls the placement of a predefined category of cells within the partial blockage.

To create a category blockage,

1. Create the cell category by defining a user attribute for the category and applying it to the affected cell references or instances.

The syntax to define the user attribute is

```
define_user_attribute
  -type boolean -class lib_cell | cell attribute_name
```

The syntax to apply the user attribute is

```
set_attribute inst_or_ref_name attribute_name true | false
```

To prevent the cell from being placed in the category blockage, set the attribute to `true`. To allow the cell in the category blockage, set the attribute to `false`.

When defining and applying attributes, observe the following rules:

- A blockage can only be controlled by a single attribute.
 - Multiple blockages can be controlled by the same attribute.
 - A cell can have multiple attributes, which could impact its placement in multiple category blockages.
 - The same attribute can be applied to a reference and an instance.
 - If a cell instance and its reference have different attributes, the attribute on the cell instance takes precedence.
2. Create the category blockage by using the `create_placement_blockage` command with the `-type partial`, `-blocked_percentage`, and `-category` options. The argument to the `-category` option is the name of the user attribute.

```
create_placement_blockage -type partial \
  -blocked_percentage percent -category attribute_name
```

For example, to define a blockage that prevents a certain category of cells from being placed within it, use the following commands:

```
icc_shell> define_user_attribute -type boolean \
    -class lib_cell dr_cells
icc_shell> set_attribute [get_lib_cell mylib/dr*] dr_cells true
icc_shell> create_placement_blockage -bbox {4300 2400 4500 2700} \
    -type partial -blocked_percentage 45 -category dr_cell
```

For example, to define a blockage that allows only a certain category of cells to be placed within it, use the following commands:

```
icc_shell> define_user_attribute -type boolean \
    -class lib_cell non_iso
icc_shell> set_attribute [get_lib_cell mylib/*] non_iso true
icc_shell> set_attribute [get_lib_cell mylib/iso*] non_iso false
icc_shell> create_placement_blockage -bbox {4300 2400 4500 2700} \
    -type partial -blocked_percentage 30 -category non_iso
```

Defining a Pin Blockage

To define a pin blockage, specify the bounding box, type (`-no_pin` option), and optionally the name for the placement blockage. By default, all layers are blocked for global routing and pin assignment. To specify which layers are blocked, use the `-blocked_layers` option.

Querying Placement Blockages

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

Removing Placement Blockages

To remove placement blockages from the design, use the `remove_placement_blockage` command. To remove all placement blockages from the design, use the `-all` option. To remove specific placement blockages from the design, specify the placement blockage names.

Creating Placement Blockages for Thin Channels

Thin channels often occur at top-level floorplans. Placing cells of the correct size in the thin channels can prevent large cell displacement caused by high local density after optimization. You can create placement blockages that fit in the thin channels to improve the quality of results at the top level by using the `derive_placement_blockages` command.

By default, the `derive_placement_blockages` command derives hard placement blockages to prevent the tool from placing standard cells in the thin channels. This command does not support soft and partial placement blockages. To specify the threshold value of the channel width, use the `-thin_channel_width` option. If you do not specify this option, the command uses the largest buffer width in the target libraries. To add the derived placement blockages

to the design, use the `-apply` option. To generate a Tcl script for the derived placement blockages, use the `-output_script` option.

For example, the following command derives hard placement blockages and adds them to the design.

```
icc_shell> derive_placement_blockages -output_script test.tcl -apply
```

To change the type of blockages, you can use the following commands to generate a script and then change the setting of the `-type` option of the `create_placement_blockage` command in the script. Note that the names of the placement blockages have the `ICC_THIN_CHAN_PLACE_BLKG_` prefix.

```
icc_shell> open_mw_cel init_cel
icc_shell> derive_placement_blockages -output_script floorplan_bkg.tcl
icc_shell> source floorplan_bkg.tcl
```

The following script shows an example of an output file:

```
create_placement_blockage -name ICC_THIN_CHAN_PLACE_BLKG_#0 -type hard \
    -bbox { 425.565 815.215 883.790 823.215}
create_placement_blockage -name ICC_THIN_CHAN_PLACE_BLKG_#1 -type hard \
    -bbox { 425.565 1596.260 883.790 1604.375}
create_placement_blockage -name ICC_THIN_CHAN_PLACE_BLKG_#2 -type hard \
    -bbox { 883.790 815.215 891.790 910.215}
create_placement_blockage -name ICC_THIN_CHAN_PLACE_BLKG_#3 -type hard \
    -bbox { 905.425 1072.665 1079.585 1087.335}
create_placement_blockage -name ICC_THIN_CHAN_PLACE_BLKG_#4 -type hard \
    -bbox { 1082.935 1604.375 1089.515 2108.205}
create_placement_blockage -name ICC_THIN_CHAN_PLACE_BLKG_#5 -type hard \
    -bbox { 1270.930 425.565 1278.930 878.900}
```

Setting the Congestion Options

Congestion occurs when the number of wires going through a region exceeds the capacity of that region. This condition is detected by global routing. When you use the `place_opt -congestion` command, the IC Compiler tool attempts to minimize congestion during placement and optimization.

When you enable congestion minimization, the tool runs the Zroute global router during placement and optimization with the following options:

```
set_route_zrt_global_options -effort low \
    -timing_driven false -crosstalk_driven false
```

Note:

If you use the `set_route_zrt_global_options` command to set different values for these options, the tool ignores your settings during congestion-driven placement.

You can set certain options related to congestion estimation and minimization with the `set_congestion_options` command.

- To specify how densely cells can be packed in uncongested regions to relieve congestion in other regions, use the `-max_util` option. Set this option based on how much area is needed for placement. The default maximum utilization is 0.95.
- To specify the availability of routing resources for a layer, use the `-layer` and `-availability` options. For example, in a design whose M5 and M6 layers will be 70 percent occupied by future power and ground routing, set the availability of M5 and M6 to 0.30.

By default, the congestion option settings apply to the whole design. To apply the settings only to a specific region of the design, use the `-coordinate` option to specify the lower-left and upper-right coordinates of the region's bounding box.

To report congestion option settings, use the `report_congestion_options` command. To remove congestion option settings, use the `remove_congestion_options` commands.

To visualize congestion conditions, use the global route congestion visual mode as described in “[Congestion Maps](#)” on page 4-73. To report congestion conditions, including DRC violations and routing density, use the `report_congestion` command. By default, the command reports the congestion conditions based on global routing. You can also report the congestion conditions based on track assignment or detail routing by setting the `-routing_stage` option to `track` or `detail` respectively. For more details, see “[Analyzing Congestion](#)” on page 6-114.

Controlling Placement Density

For placement that is not set up for congestion removal, you can control how densely cells can be packed by using the `placer_max_cell_density_threshold` variable. Set this variable to a value between one and the overall average utilization of your design. Choosing a value near one allows cells to clump together more densely. A value of one allows no gaps between cells. You should choose a high value for designs with low utilization to improve timing and a low value for congested designs to avoid cell clumping. By default, this variable is set to -1 to disable this feature.

For example, if the global utilization is 40 percent, choose a value between 0.4 and 1.

```
icc_shell> set_app_var placer_max_cell_density_threshold 0.6
```

If you see a high density of cells in thin channels or around macros and placement blockages, you can improve the cell placement density by using the following variable setting before you run any placement-based command:

```
icc_shell> set_app_var placer_reduce_high_density_regions true
```

Note:

You might see an increase in runtime when you set the `placer_reduce_high_density_regions` variable to `true`. You might also see an increase in wire length and a degradation of the timing QoR of your design.

The IC Compiler tool can identify narrow placement areas as channels and treat these channels differently during coarse placement. To enable this feature, use the `placer_channel_detect_mode` variable.

If you set the `placer_channel_detect_mode` variable to

- `true`, the tool identifies channels and reduces the maximum cell density allowed in the channels during coarse placement. This helps reduce congestion.
- `auto`, the tool identifies the channels and if the number of channels in the design is minimal, the tool does not treat them any differently. However, if the number of channels in the design is substantial, the tool reduces the maximum cell density allowed in the channels during coarse placement. If the design has a substantial number of channels, setting this variable to `auto` can increase runtime and memory usage.
- `false`, the default, the tool does not treat channel areas differently.

Defining Placement Bounds

A placement bound is a constraint that controls the placement of groups of leaf cells and hierarchical cells. It allows you to group cells to minimize wire length and place the cells at the most appropriate locations. For example, you might want to define a bound for clock-gating cells or extremely timing-critical groups of cells that you do not want disrupted by the placement of other logic.

To define a placement bound, use the `create_bounds` command. When you define a bound, you must use the `-name` option to specify its name. By default, the command creates a soft bound. To create a hard bound, use the `-type hard` option. To create an exclusive bound, use the `-exclusive` option.

You must also specify the options required for the specific type of bound you want to create. For details about creating the various types of move bounds, see

- [Defining Move Bounds](#)
- [Defining Group Bounds](#)

Defining Move Bounds

A move bound is a fixed region within which to place a set of cells. It is comprised of one or more rectangular or rectilinear shapes, which can be abutted, disjoint, or overlapping.

To define the boundaries of these shapes, use the `-coordinate` option with the `create_bounds` command.

- To specify the boundary of a rectangle shape, use the following format to specify its lower-left and upper-right corners:

```
{llx1 lly1 urx1 ury1}
```

- To specify the boundary of a rectilinear shape, use the following format to specify the lower-left and upper-right corners of the individual rectangles that make up the rectilinear shape:

```
{llx1 lly1 urx1 ury1 llx2 lly2 urx2 ury2 ...}
```

The tool supports the following types of move bounds:

- Soft move bounds

For soft move bounds, the tool tries to place the cells within the specified region; however, there is no guarantee that the cells are placed inside the bounds.

To define a soft move bound, use the following syntax:

```
create_bounds -name name -coordinate {coordinates} [-type soft]
[bound_objects]
```

For example, to create a rectangular soft move bound for the INST_1 cell instance with its lower-left corner at (100, 100) and its upper-right corner at (200, 200), use the following command:

```
icc_shell> create_bounds -name b1 -coordinate {100 100 200 200} INST_1
```

- Hard move bounds

For hard move bounds, the tool must place the cells within the specified region.

To define a hard move bound, use the following syntax:

```
create_bounds -name name -coordinate {coordinates} -type hard
[bound_objects]
```

- Exclusive move bounds

For exclusive move bounds, the tool must place the cells within the specified region and must place all other cells outside of the region.

To define an exclusive move bound, use the following syntax:

```
create_bounds -name name -coordinate {coordinates} -exclusive
[bound_objects]
```

Defining Group Bounds

A group bound is a floating region within which to place a set of cells. The tool supports the following types of group bounds:

- Soft group bound

For soft group bounds, you specify the width and height of the rectangles that comprise the bound. The tool determines the best location for the group bound and tries to place the objects within the specified dimensions; however, there is no guarantee that the cells are placed inside the bounds.

To define a soft group bound, use the following syntax:

```
create_bounds -name name -dimension {height width} [-type soft]
[bound_objects]
```

For example, to create a soft group bound for the INST_1 and INST_2 cell instance with a width of 100 and a height of 100, use the following command:

```
icc_shell> create_bounds -name b2 -type soft -dimension {100 100} \
{INST_1 INST_2}
```

- Hard group bound

For hard group bounds, you specify the width and height of the rectangles that comprise the bound. The tool determines the best location for the group bound and must place the objects within the specified dimensions.

To define a hard group bound, use the following syntax:

```
create_bounds -name name -dimension {height width} -type hard
[bound_objects]
```

- Dimensionless group bound

For dimensionless group bounds, you specify the effort used to bring the objects closer together inside the group bound. The tool determines the shape and location of the group bound. The default effort is medium; you can also specify low, high, or ultra.

To define a dimensionless group bound, use the following syntax:

```
create_bounds -name name -effort effort_level
[bound_objects]
```

For example, to create a dimensionless group bound in which the tool uses a high level of effort to place the INST_3 and INST_4 cell instance closer within the group bound, use the following command:

```
icc_shell> create_bounds -name b3 -effort high {INST_3 INST_4}
```

- Diamond bound

A diamond bound constrains the placement of cells to be within a specified Manhattan distance from a specific cell, port, or pin. You specify the central object and the maximum distance. The tool determines the best location of the diamond bound and tries to place the cells within the constraints.

To define a diamond bound, use the following syntax:

```
create_bounds [-name bound_name]  
-diamond object -dimension distance {cell_list}
```

For example, to create a diamond bound to constrain the placement of the INST_5 cell instance such that it is within a 50-micron Manhattan distance from the DATA_IN_1 port, use the following command:

```
icc_shell> create_bounds -name b4-dimension 50 \  
-diamond [get_ports DATA_IN_1] [get_cell INST_5]
```

Updating, Querying, and Removing Placement Bounds

To add cells to or remove cells from an existing bound, use the `update_bounds` command.

To report the placement bounds in your design, use the `report_bounds` command.

To return a collection of placement bounds in the current design that match certain criteria, use the `get_bounds` command.

To remove placement bounds from your design, use the `remove_bounds` command.

Preventing Cell Placement in the Default Voltage Area

Certain multivoltage design styles implement abutted voltage areas at the top level, so no space is allowed for cell placement in the default voltage area. You can prevent the insertion of new cells in the default voltage area by setting the `mv_no_cells_at_default_va` variable to `true`. The default for this variable is `false`. When this variable is set to `true`, the tool inserts new cells in the lower-level voltage areas, rather than the default voltage area. It also eliminates the requirement for setting the `dont_touch` attribute on the affected top-level nets in the default voltage area. This variable is effective for designs with both abutted and nonabutted voltage areas at the top level.

For more information about the default voltage area, see “Default Voltage Area” on page 12-17.

Defining Spacing Requirements

The IC Compiler tool places standard cells by using a discrete grid, which is defined by the unit tile. By default, there are no spacing constraints between cells during placement. To enhance yield, you can define the valid spacing between standard cells in the same row.

Intercell spacing rules are implemented by attaching labels, which are similar to attributes, to the left and right sides of library cells, assuming that the cell is in its north orientation, and specifying the invalid spacings between these labels.

To define the intercell spacing rules,

1. Add labels to the right and left sides of library cells that have minimum spacing constraints by using the `set_lib_cell_spacing_label` command.

Use the `-name` option to specify the label names. You can specify one or more labels. The labels are assigned to the right side of the cells specified with the `-right_lib_cells lib_cells` option and to the left side of the cells specified with the `-left_lib_cells lib_cells` option. You can assign the same label to both the right and left sides of a cell by including it in both the `-right_lib_cells` and `-left_lib_cells` options.

For example, to assign a label named X to the right side of the cellA library cell and the left side of the cellB and cellC library cells, enter the following command:

```
icc_shell> set_lib_cell_spacing_label -names {X} \
           -right_lib_cells {cellA} -left_lib_cells {cellB cellC}
```

To assign labels named Y and Z to the right side of the cellB library cell, enter the following command:

```
icc_shell> set_lib_cell_spacing_label -names {Y Z} \
           -right_lib_cells {cellB}
```

2. Define the spacing requirements between the labels by using the `set_spacing_label_rule` command.

Use the `-labels` option to specify which labels are being constrained. You must specify exactly two labels, which can be the same or different labels. Specify the range of invalid spacings, in terms of the unit tile, as the command argument.

For example, to specify that labels X and Y cannot abut (there must be at least one unit tile between them), enter the following command:

```
icc_shell> set_spacing_label_rule -labels {X Y} {0 0}
```

To specify that two X labels cannot have a spacing of two unit tiles, enter the following command:

```
icc_shell> set_spacing_label_rule -labels {X X} {2 2}
```

To specify that labels X and Z must have a spacing of less than two unit tiles or more than four unit tiles, enter the following command:

```
icc_shell> set_spacing_label_rule -labels {X Z} {2 4}
```

To define the boundary spacing rules,

1. The tool assigns a built-in label named SNPS_BOUNDARY to chip boundaries that include chip core boundary (two ends of a row), fixed hard macro boundary, and hard placement blockage boundary. You cannot use the `set_lib_cell_spacing_label` command to assign the built-in label to library cells.
2. Define the spacing requirements between the built-in label and other labels by using the `set_spacing_label_rule` command.

For example, to specify that the label X referenced cells should not be placed within zero to one unit tile from the chip boundaries, enter the following commands:

```
icc_shell> set_lib_cell_spacing_label -names {X} \
    -right_lib_cells {AND*} -left_lib_cells {AND*}
icc_shell> set_spacing_label_rule -labels {X SNPS_BOUNDARY} {0 1}
```

To report both the intercell and boundary spacing rules, use the `report_spacing_rules` command with the `-all` option. To report the intercell spacing rules for a specific collection of library cells, use the `report_spacing_rules` command with the `-of_library_cells` option.

To remove all spacing rules, use the `remove_all_spacing_rules` command.

Note:

If the power or ground net is defined as a complete blockage, the legalizer and the `check_legality` command ignore the spacing rule violations between library cells and the power or ground net. If the power or ground net is defined as a partial blockage, the legalizer and the `check_legality` command check for spacing rule violations between library cells and the power or ground net.

Defining Target Library Subsets

The IC Compiler tool allows you to restrict optimization using a subset of the target library. Normally, optimization can select any library cell from the target library. However, you can use the `set_target_library_subset` command to restrict the library cells to be used in a particular hierarchical cell or in the top-level design. For example, you can omit specific double-height cells in some hierarchical cells even though they are mixed with other needed library cells in the target library. The subset restriction applies only to cell instances that are created or modified during optimization.

When you use the `set_target_library_subset` command on a hierarchical cell or the top-level design, the subset applies to all lower cells in the hierarchy, except for those cells

that have a different library subset constraint explicitly set on them. A subset at a lower level overrides any subset specified at a higher level. The tool does not incrementally refine the upper subset.

The `set_target_library_subset` command syntax is

```
set_target_library_subset -object_list hierarchical_cells | -top
    [{library_list}] [-dont_use lib_cells] [-only_here lib_cells]
```

where

- `-object_list` or `-top` specifies where the target library subset is used. To use the target library subset on one or more hierarchical cells, specify the cells with the `-object_list` option. To use the target library subset on the top-level design, specify the `-top` option.
- `-milkyway_reflibs` is a space-separated list of Milkyway reference library paths to associate with the specified hierarchical cells. When you specify multiple reference library paths, the list must be a subset of the reference library list for the design.
- `library_list` is a space-separated list of target library file names, all of which must also be listed in the `target_library` variable. If you do not specify this option, all libraries listed in the `target_library` variable are used.
- `-dont_use lib_cells` specifies library cells that cannot be used for optimization even if they are inside the libraries specified by the `library_list` argument. You can specify the library cells by name or by collection in a space-separated list. You can also use a wildcard (*).
- `-only_here lib_cells` specifies library cells that can be used only for optimization in the specified hierarchical cells or in another hierarchical cell where the library cells have been specified by the `-only_here` option. This essentially applies a `-dont_use` option on these cells for all other target library subsets in the design, including the target library subset at the top level, unless the subsets also contain an `-only_here` option for these cells.

You can specify the library cells by name or by collection in a space-separated list. You can also use a wildcard (*).

If the `-dont_use` cell list and the `-only_here` cell list include the same cell name after wildcard expansion, the `-only_here` cell list takes precedence and the tool issues an information message.

If you specify a library name with the `-dont_use` and `-only_here` options, the tool ignores the specified library and reports an information message. For example, if you run the following command, the tool issues a message saying that library name `lib1` is irrelevant and is ignored:

```
set_target_library_subset -top -dont_use {AN2 */OR* lib1/NOR2}
```

In this case, the tool restricts the use of all library cells named NOR2.

You cannot override a `dont_use` attribute if it is set in a library. If the `dont_use` attribute is set on a target library cell, it cannot be used even if listed in the `-only_here` cell list.

IC Compiler issues a warning if you apply the `-only_here` option to a target library cell that has a `dont_use` attribute set on it. If you want to make target library cells selectively available for use with the `set_target_library_subset` command, you must first remove the `dont_use` attribute from the library with the `remove_attribute` command.

Reports that are generated by reporting commands, such as `report_cell` and `report_timing`, include a `td` attribute, indicating that a cell specified with the `-dont_use` option or the `-only_here` option is used somewhere in the design.

You cannot ungroup a subdesign if the `set_target_library_subset` command is set on it. Similarly, the tool's automatic ungrouping capability cannot ungroup the subdesign.

To check for errors and conflicts introduced by the target library subset, use the `check_mv_design -target_library_subset` command. This command checks for the following conditions:

- Conflicts between target library subsets and the global `target_library` variable
- Conflicts between operating conditions and target library subsets
- Conflicts between the library cell of mapped cells and target library subsets

To report which target library subsets have been defined both for the hierarchical cells and at the top level, use the `report_target_library_subset` command with the appropriate library cell list.

To remove a target library subset constraint from the design or a hierarchy instance, use the `remove_target_library_subset` command with the appropriate library list and cell list.

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 might 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 does not degrade the quality of results (QoR).

Setting the Preferred Buffers for Hold Fixing

You can specify a list of preferred buffers to fix hold violations during preroute optimization by using the `set_prefer` and `set_fix_hold_options` commands. For example, the following commands instruct the tool to use cells BUF1 and BUF2 as the preferred cells during hold fixing. Note that cells BUF1 and BUF2 can also be used to resolve setup and DRC violations.

```
icc_shell> set_prefer -min {BUF1 BUF2}
icc_shell> set_fix_hold_options -preferred_buffer
icc_shell> psynopt -congestion -area_recovery
```

If you use the `set_dont_use` command to set the `dont_use` attribute on cells BUF1 and BUF2 as shown in the following script, the tool uses cells BUF1 and BUF2 to fix hold violations, but not setup and DRC violations.

Enabling Tie Cell Insertion

A tie cell is a special-purpose standard cell whose output is constant high or constant low and is used to hold the input of another cell at the given constant value. To prepare the `place_opt` or `psynopt` commands for automatic insertion and optimization of tie-offs required in the design, execute commands similar to the following:

```
set_auto_disable_drc_nets -constant false
set_app_var physopt_new_fix_constants true
set_attribute [...] max_fanout 12
set_attribute [...] max_capacitance 0.2 -type float
```

Optimization means using a single tie cell to hold as many inputs as possible at a given logic level, while meeting specified maximum fanout and maximum capacitance constraints. The `set_auto_disable_drc_nets` command enables DRC fixing on constant nets. Setting the `physopt_new_fix_constants` variable to `true` causes the tool to observe the maximum capacitance constraint during tie-off optimization. The maximum capacitance constraint is determined by the `max_capacitance` attribute, which is set with the `set_max_capacitance` or `set_attribute` command. The `set_attribute` command is used to specify explicitly both the maximum fanout and maximum capacitance constraints for objects in the design.

By default, the tool performs tie-off optimization one hierarchical level at a time. You can enable tie-off optimization down to the lowest hierarchy without port punching by setting the `tieoff_hierarchy_opt` variable to `true`, changing it from its default of `false`. When this variable is set to `true`, the tool fixes DRC violations of constant nets crossing multiple hierarchies and removes the driving cells of these constant nets. To keep these driving cells, set the `tieoff_hierarchy_opt_keep_driver` variable to `true`, changing it from its default of `false`. The tool does not insert tie cells to dangling hierarchical ports.

You can also insert tie cells manually with the `connect_tie_cells` command. The command inserts tie cells and connects them to specified cell ports, while meeting the maximum fanout and maximum wire length constraints specified in the command.

Preserving `dont_touch` Nets During Optimization

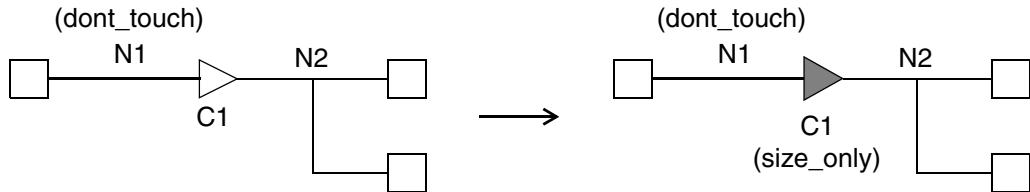
By default, the IC Compiler tool ignores the `dont_touch` attribute on a net that is connected to a cell without the `dont_touch` attribute and can split the net during optimization. To honor the `dont_touch` attribute on this type of net, you specify the following command:

```
icc_shell> set_dont_touch_network -no_propagate object_list
```

The object list a list of one or more clocks, ports, or pins in the design. The command prevents cells and nets in the transitive fanout of the specified object from being replaced or modified during optimization. The `-no_propagate` option prevents the don't touch network from being propagated through logic gates.

In the example circuit shown in [Figure 4-3](#), N1 is a net marked with the `dont_touch` attribute, but the net driver does not have the `dont_touch` attribute. By default, net N1 can be split during optimization. However, if you apply the `set_dont_touch_network` command to the pin driving net N1 and use the `-no_propagate` option, the tool marks the C1 cell with the `size_only` attribute and leaves net N1 untouched during optimization.

Figure 4-3 dont_touch Net Remains Untouched During Optimization



Limits on Fanout During Optimization

During the area recovery phase of optimization, the IC Compiler tool removes cells on noncritical timing paths, which increases the fanout of the corresponding nets. Paths which are timing-noncritical during initial optimization can become timing-critical during postroute optimization to correct signal integrity issues.

To avoid removing cells from potential timing-critical paths, you can specify a maximum fanout threshold limit for optimization by setting the `psynopt_high_fanout_legality_limit` variable. IC Compiler does not perform any optimization that results in a fanout greater than the value specified for this variable. By default, this variable has a value of 40.

Setting Placement and Optimization Attributes

Table 4-1 lists the restrictions imposed on the placement and optimization of cells by attributes.

Table 4-1 Restrictions Imposed by Placement and Optimization Attributes

Attribute	Coarse placement	Detail placement	Optimization
<code>is_fixed</code>	Cannot move cells	Cannot move cells	Cannot move, rotate, or resize cells
<code>cts_fixed</code> (imposed on clock buffers by clock tree synthesis)	Cannot move cells	Cannot move cells	Cannot move, rotate, or resize cells
<code>is_soft_fixed</code>	Cannot move cells	No restrictions	No restrictions
<code>size_only</code>	No restrictions	No restrictions	Can only resize cells
<code>in_place_size_only</code>	Cannot move cells	No restrictions	Can resize cells only if there is room
<code>cts_in_place_size_only</code> (imposed on clock sinks by clock tree synthesis)	Cannot move cells	No restrictions	Can resize cells only if there is room
<code>dont_touch</code>	No restrictions	No restrictions	Cannot move, rotate, or resize cells

To report information about size-only cells, including the reasons for the size-only setting, use the `report_size_only` command. You can generate a report for all the cells in the design, a collection of cells, or a specific cell instance.

Preparing for High-Fanout Net Synthesis

During placement and optimization, the IC Compiler tool does not buffer clock nets as defined by the `create_clock` command, but it does, by default, buffer other high-fanout nets, such as resets or scan enables, using a built-in high-fanout synthesis engine.

Before running high-fanout net synthesis during the `place_opt` step, define the buffering options by using the `set_ahfs_options` command.

During high-fanout net synthesis, the tool automatically analyzes the buffer trees to determine the fanout thresholds by default, and then it removes and builds buffer trees.

The high-fanout synthesis engine does not buffer nets that are set as ideal nets or nets that are disabled with respect to design rule constraints.

As an alternative to performing high-fanout net synthesis during the `place_opt` command, you can use the `create_buffer_tree` command to run standalone high-fanout net synthesis. The command removes the preexisting buffer trees in the design and re-creates the buffer trees. To preserve the preexisting buffer trees, you can run incremental high-fanout net synthesis by specifying the `-incremental` option.

To get information about the buffer trees in your design, use the `report_buffer_tree` command.

To remove buffer trees from your design, use the `remove_buffer_tree` command.

Specifying Layer Optimization Settings

During preroute optimization, layer optimization takes into consideration any layer constraints and nondefault routing rules specified on nets. This increases the accuracy of net delay estimation during preroute optimization.

When you run the `place_opt` command, the tool performs layer optimization by default. You can disable it by using the `set_place_opt_strategy -layer_optimization false` command. To check if layer optimization has been disabled, use the `report_place_opt_strategy` command.

Before you run the `place_opt` command, you can create net search patterns and define layer constraints and nondefault routing rules for the nets that match these search patterns. During optimization, the tool creates collections of nets based on these search patterns and applies the appropriate layer constraints and nondefault routing rules to these collections of

nets. For more information about creating net search patterns and defining layer constraints, see “[Creating Layer Constraints](#)” on page 4-25.

During layer optimization, the tool automatically assigns minimum and maximum routing layers for long nets that are on timing-critical paths in the following situations:

- If you do not specify any net search patterns or the associated layer constraints.
- If you specify net search patterns and the associated layer constraints, but none of the nets in the design match the search patterns you specify.

By default, the tool uses virtual-route-based congestion estimation when deriving layer constraints for critical nets. To use global-route-based congestion estimation instead, specify the `-layer_optimization_effort high` option with the `set_place_opt_strategy` command.

To use more routing information to drive preroute optimization, specify the `-consider_routing true` option with the `set_place_opt_strategy` command. Then, when you run the `place_opt` command, the tool performs global routing, track assignment, and post route extraction to generate RC models that are used during subsequent preroute optimization performed by running the `place_opt`, `psynopt`, `preroute_focal_opt`, or `clock_opt -only_psyn` command. The RC models are generated based on the current scenario, so specify the most critical scenario as the current scenario before you run the `place_opt` command.

Note:

If you have existing net search patterns and layer constraints, running the `place_opt` command with the `set_place_opt_strategy -consider_routing true` setting removes all of your existing net search patterns and layer constraints.

Creating Layer Constraints

Use the `create_net_search_pattern` command to select nets based on the following criteria: fanout, bounding box perimeter, net length, blocked area ratio, aspect ratio, and setup slack.

For example, to create a search pattern that includes all nets with a fanout larger than or equal to 100, use the following command:

```
icc_shell> create_net_search_pattern -fanout_lower_limit 100
```

When you create a search pattern, the tool automatically assigns it an identification number, starting with 1, and returns the identification number. As you create more search patterns, the tool increments the identification number by 1.

The net search pattern created in the following example is the fourth search pattern; as a result, the tool assigns it an identification number of 4.

```
icc_shell> create_net_search_pattern -net_length_lower_limit 500
Information: Assigning pattern id to be 4. (PAT-003)
4
```

The identification number is required when specifying layer constraints and nondefault routing rules for nets that match the pattern. To save the identification number that the tool returns when you create a search pattern, use the `set_app_var` command. For example, to create a search pattern that includes all nets with a fanout larger than or equal to 1000, and to save its identification number, use the following command:

```
icc_shell> set_app_var pattern_id_no \
[create_net_search_pattern -net_length_lower_limit 1000]
```

To generate a report on a search pattern and the associated search criteria, use the `report_net_search_pattern` command, as shown in the following example:

```
icc_shell> report_net_search_pattern -pattern $pattern_id_no
```

To remove a search pattern, use the `remove_net_search_pattern` command, as shown in the following example:

```
icc_shell> remove_net_search_pattern -pattern $pattern_id_no
```

To get a collection of nets that match a specific search pattern, use the `get_matching_nets_for_pattern` command, as shown in the following example:

```
icc_shell> get_matching_nets_for_pattern -pattern $pattern_id_no
```

Some nets that match a search pattern might not be optimizable. For example, a net in the collection might be an ideal net, and as a result it is not optimizable. To get all optimizable nets that match a search pattern, use the `-optimizable` option of the `get_matching_nets_for_pattern` command, as shown in the following example:

```
icc_shell> get_matching_nets_for_pattern -pattern $pattern_id_no \
-optimizable
```

To specify layer constraints and nondefault routing rules for the collection of nets that match a search pattern, use the `set_net_search_pattern_delay_estimation_options` command. For example, to specify that nets matching a certain search pattern are restricted to layers M7, M8, and M9, use the following command:

```
icc_shell> set_net_search_pattern_delay_estimation_options \
-pattern $pattern_id_no -min_layer_name M7 -max_layer_name M9
```

To report the layer constraints and nondefault routing rules specified for a search pattern, use the `report_net_search_pattern_delay_estimation_options` command, as shown in the following example:

```
icc_shell> report_net_search_pattern_delay_estimation_options \
           -pattern $pattern_id_no
```

When multiple search patterns are created, a single net could match multiple patterns. By default, the tool prioritizes the search patterns based on the order in which they are created. A search pattern created first has a higher priority over a search pattern created subsequently. Because the tool always begins numbering search patterns with 1, if five search patterns are created, the default priority among the five search patterns would be 1 2 3 4 5. To change the default priority, use the `set_net_search_pattern_priority` command as shown in the following example:

```
icc_shell> set_net_search_pattern_priority "4 5 1 2 3"
```

To report the search pattern priority, use the `report_net_search_pattern_priority` command.

Note:

The methodology described in this section for creating collections of nets and specifying layer constraints and nondefault routing rules applies only to layer optimization which is performed when you run the `place_opt` command. It does not apply to layer optimization enabled by using the `preroute_focal_opt -layer_optimization` command. For more information about performing layer optimization by using the `preroute_focal_opt -layer_optimization` command, see “[Performing Layer Optimization on Existing Buffer Trees](#)” on page 4-89.

Preparing for Clock Tree Synthesis During Placement

If your design contains simple clock tree structures and uses the same design rule constraints for placement and clock tree synthesis, you can simplify the design flow by performing clock tree synthesis and optimization during placement.

To perform clock tree synthesis and optimization during placement,

1. Define the clock trees as described in [Chapter 5, “Clock Tree Synthesis”](#).
2. Specify the clock tree synthesis options by using the `set_place_opt_cts_strategy` command, as shown in [Table 4-2](#)

Table 4-2 The set_place_opt_cts_strategy Options

To do this	Use this option
Specify the operating condition to use for clock tree synthesis and optimization. If you do not specify this option, the maximum operating condition is used.	<code>-operating_condition min max min_max</code>
Prevent routing of the clock tree.	<code>-no_clock_route</code>
Perform interclock delay balancing.	<code>-inter_clock_balance</code>
Fix hold time violations on the clock tree.	<code>-fix_hold</code>
Update the clock latency values.	<code>-update_clock_latency</code>

3. Use the `-cts` option when you run the `place_opt` command.

Inserting Port Protection Diodes

The IC Compiler tool can automatically insert protection diodes on subdesign ports to prevent antenna violations at the top level. You insert the port protection diodes after floorplanning but before starting placement. Use the `insert_port_protection_diodes` command to add diodes to the specified ports to your netlist, one diode per port. The new diodes are legalized near their corresponding ports. Note the following limitations:

- This command does not add diodes to ports with the `dont_touch` attribute.
- This command is not voltage-area aware. You need to select correct voltage diodes for ports in different voltage areas.

To report the port protection diodes that are inserted in your design, use the `report_port_protection_diodes` command.

Setting Up Multithreading

By default, the `place_opt` command uses a single thread to perform placement and optimization. If the placement step takes up a large portion of the `place_opt` runtime, you can enable the multithreaded placer. When working with multicorner-multimode designs, you can reduce the elapsed time required to run the `place_opt` command by using multithreading for preroute timing analysis.

To enable multithreading, use the `set_host_options` command with the `-max_cores` option. You can limit the number of threads used for placement by setting the `placer_max_parallel_computations` variable. To limit the number of threads used for multicorner-multimode timing analysis, set the `timing_max_parallel_computations` variable. For more information about using the `set_host_options` command to configure multithreading, see “[Enabling Multicore Processing](#)” on page 2-12.

Performing Placement and Optimization

After you have finished design planning and power planning, you can perform placement and optimization on your design.

To perform placement and optimization, use the `place_opt` command or choose Placement > Core Placement and Optimization in the GUI.

The `place_opt` command performs coarse placement, high-fanout net synthesis, physical optimization, and legalization. In addition, it can perform clock tree synthesis, scan chain reordering, and power optimization.

Table 4-3 describes the available `place_opt` options.

Table 4-3 The place_opt Options

GUI object	Command option	Description
Effort option	<code>-effort low medium high</code>	Improves the quality of results or reduce runtime. Higher effort levels use additional runtime in an attempt to improve the quality of results. The default is medium.
“Recover area” check box	<code>-area_recovery</code>	Recovers area for cells not on timing-critical paths.
“Power optimization” check box	<code>-power</code>	Enables leakage-power optimization and power-aware timing optimization. For multicorner-multimode designs, you must use the <code>set_scenario_options</code> command to identify the leakage scenarios before enabling leakage-power optimization and power-aware timing optimization.
“Clock compilation, optimization and routing” check box	<code>-cts</code>	Runs clock tree synthesis and optimization with placement and optimization. Typically, this is used for the quick flow for designs not requiring extensive clock tree synthesis and optimization work. For more information about clock tree synthesis and optimization, see Chapter 5, “Clock Tree Synthesis.”
“Reorder scan during placement” check box	<code>-optimize_dft</code>	Reorders scan chains.
N/A	<code>-optimize_icgs</code>	Optimizes clock-gating logic. For more information, see “Optimizing Clock-Gating Logic” on page 4-38 .
“Consider scan connections during placement” check box	<code>-consider_scan</code>	Considers the scan chain connections during placement. By default, the command ignores scan nets.

Table 4-3 The place_opt Options (Continued)

GUI object	Command option	Description
“Continue with missing scan definitions” check box	-continue_on_missing_scandef	Continues placement when the design contains scan chains but no SCANDEF data. By default, missing SCANDEF data causes the command to exit with an error message. Setting this option enables the placer to continue with a warning and results in reduced QoR.
“Minimize congestion” check box	-congestion	Reduces congestion for improved routability. For best results, use this option only for congested designs.
“Use Synopsys physical guidance” check box	-spg	Enables the tool to use the Synopsys physical guidance information in the Design Compiler tool as a starting point for place and route. For more information, see “ Using Physical Guidance From the Design Compiler Tool ” on page 4-39.
“Skip initial placement” check box	-skip_initial_placement	Omits initial placement. Do not use this option with the -spg option. For more information, see “ Omitting Initial Placement ” on page 4-39.

Note:

When you perform preroute optimization by using the `place_opt`, `place_opt_feasibility`, `psynopt`, `preroute_focal_opt`, `clock_opt -only_psyn`, or `clock_opt_feasibility -only_psyn` command, the IC Compiler tool does not fix maximum fanout violations. However, during preroute optimization, the tool fixes other logical DRC violations such as maximum capacitance and maximum transition time violations. You can fix maximum fanout violations during postroute optimization by using the `route_opt` or `focal_opt` command.

Performing Placement and Optimization on Designs With Scan Chains

During logic synthesis, sequential elements are replaced with scan cells that can be configured as serial chain-of-state elements to aid in testability. By setting certain test signals, the scan cells can either capture values from the functional combinational logic or the test data.

The order of the scan cells can be changed under some constraints without affecting the testability of the device. Because the order of the cells in a scan chain is flexible, giving functional nets higher priority during initial placement by ignoring the scan chains can improve QoR. After initial placement, you can further improve the QoR by repartitioning and reordering the scan chains based on the initial placement.

Both ignoring scan chains during initial placement and optimizing scan chains by repartitioning and reordering require annotation of the scan chain information. For information about annotating the scan chain information, see [“Annotating the Scan Chain Information” on page 3-21](#).

Note:

If your design contains scan chains, but you do not load the SCANDEF file, the placement and optimization commands issue an error message. To prevent this error message and continue with the flow, use the `-continue_on_missing_scandef` option with the command. When you use this option, the IC Compiler tool considers the scan chains during placement, which could result in reduced QoR.

Performing Initial Placement on Designs With Scan Chains

If your design contains scan chains and you have annotated the scan chain information, by default, the IC Compiler tool ignores the scan chains during initial placement. This gives the functional nets higher priority during initial placement, which can improve QoR. You can consider the scan chains during initial placement by using the `-consider_scan` option with the placement command; however, this could result in reduced QoR.

Optimizing the Scan Chains

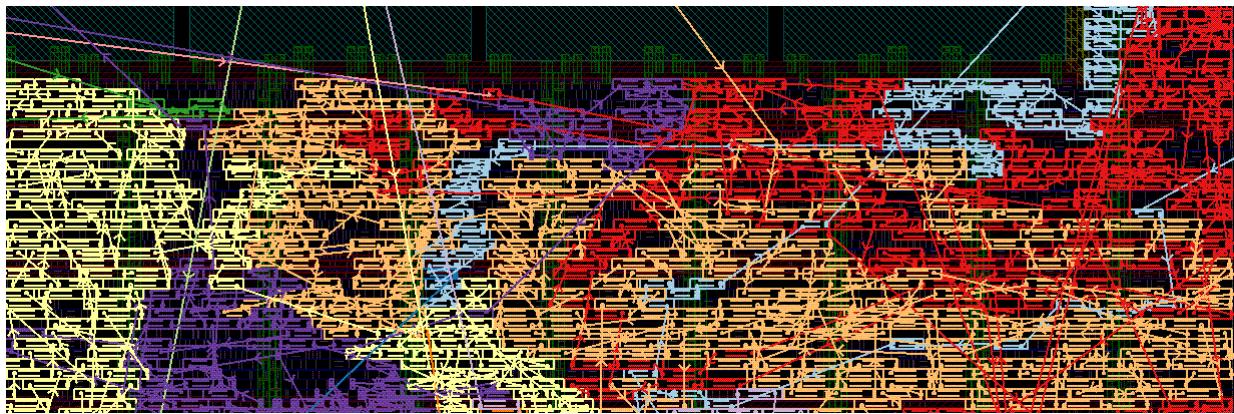
The IC Compiler tool can optimize the scan chains by repartitioning and reordering the scan chains based on the initial placement. Repartitioning scan chains involve swapping components across scan chains so that each chain contains components that are near each other. Using the placement information to repartition and reorder the scan chains provides the following benefits:

- Reduces scan chain wire length
- Minimizes congestion and improves routability

To repartition and reorder the scan chains in your design, based on scan cell locations, use the `-optimize_dft` option with the `place_opt` command. You can also choose Placement > Core Placement and Optimization and select the “Reorder scan during placement” check box in the Core Placement and Optimization dialog box in the GUI. Alternatively, you can use the `optimize_dft` standalone command to perform scan chain optimization.

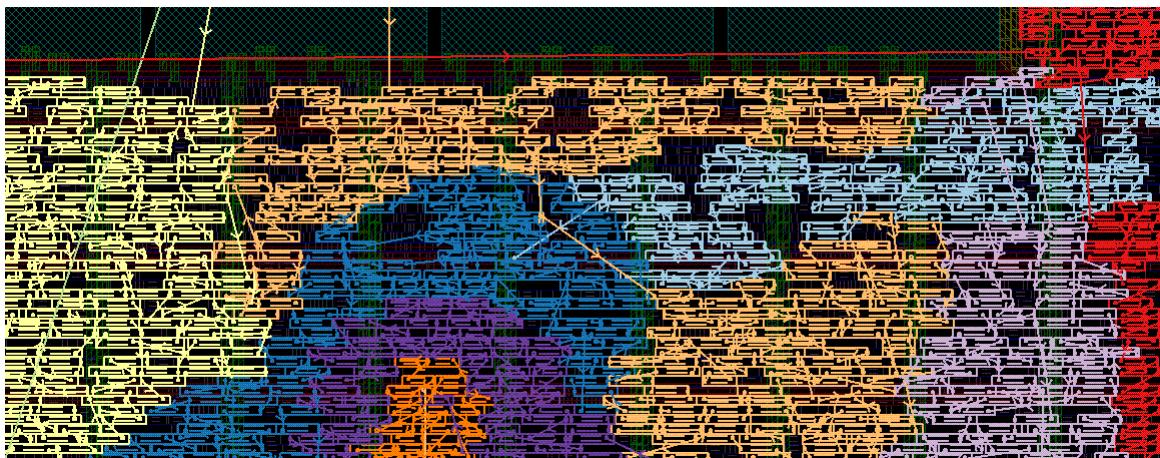
[Figure 4-4](#) shows the wiring diagram for a design containing scan chains that were synthesized by Design Compiler. Because these scan chains are not optimally ordered with respect to actual placement and clock trees, they can cause congestion and increased net delays.

Figure 4-4 Scan Chains Before Physically Aware Optimizations



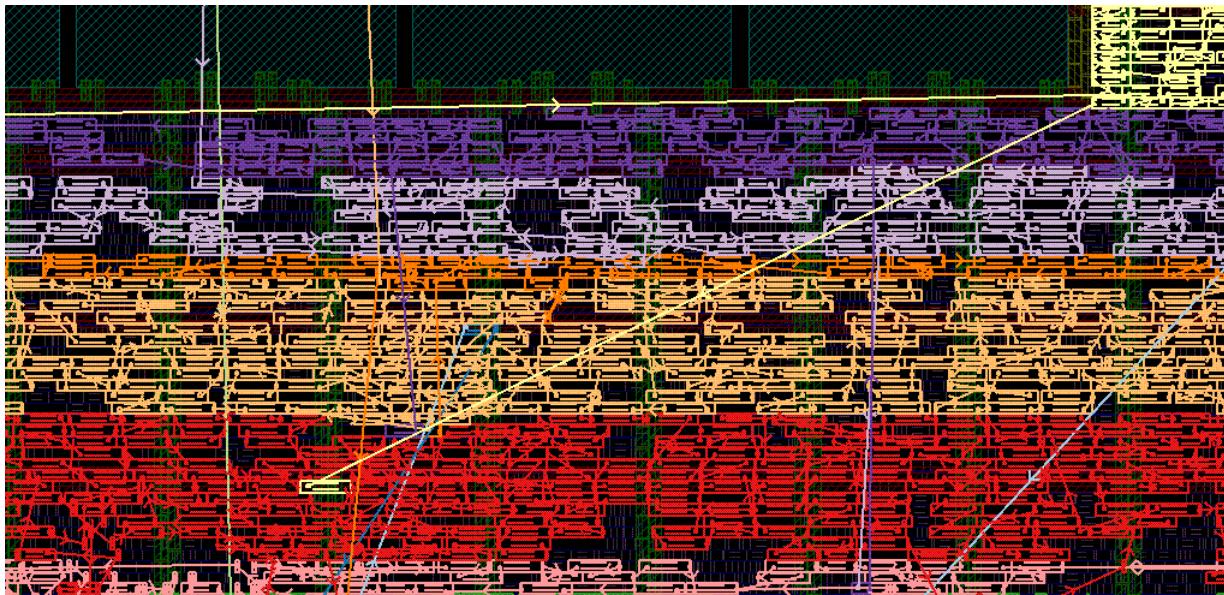
After repartitioning, the scan chains should consist of scan cells that are physically near each other, as shown in [Figure 4-5](#). This represents a design with many scan chains.

Figure 4-5 Repartitioned Scan Chains



For designs with a small number of scan chains, horizontal or vertical partitioning is done as shown in [Figure 4-6](#). If partitioning the scan chains horizontally significantly increases congestion, use the `set_optimize_dft_options -single_dir_option vertical` command to select vertical partitioning, and rerun the `place_opt` command. You can force a particular repartitioning method by using the `set_optimize_dft_options -repartitioning_method` command. You can also view the current setting by using the `report_optimize_dft_options` command. The output during DFT optimizations indicate if repartitioning was done, and, if so, what kind.

Figure 4-6 Horizontally Repartitioned Scan Chains



Following the optimizations, approximate wire lengths of the scan path wires that can be reordered are displayed. (Scan path nets in ORDERED lists cannot be reordered and therefore are not counted.) These lengths are in Milkyway database units and meant only for comparing wire lengths before and after scan chain optimizations, as shown in the following example.

```
Wire Length Before Ordering 1377917  
Wire Length After Ordering 604876
```

Performing Power Optimization

The IC Compiler tool can optimize both dynamic and static power.

- Dynamic power

This is the energy consumed because of the voltage or logic transitions in the design objects, such as cells, pins, and nets. The dynamic power consumption is directly proportional to the number and frequency of transitions in the design.

To reduce dynamic power, the tool supports low-power placement and power-aware clock tree synthesis.

- Static (leakage) power

This is the energy dissipated even when there are no transitions in the circuit. This is also known as leakage power and depends on the device characteristics. The main contributor to static power is the sub-threshold-voltage leakage in the device. At lower technology nodes, leakage power consumption contributes significantly to the total power consumption of the circuit.

Leakage power dissipation is low when the threshold voltage is high, and vice versa. Moreover, the switching delay is increased when the threshold voltage is high. So, using high-threshold-voltage cells to reduce the leakage can violate the timing constraints of the design. If the logic library supports cells with multiple threshold voltages, using cells with a lower threshold voltage for timing-critical paths and cells with a higher threshold voltage for other paths can reduce the leakage power without violating the delay.

When you enable power optimization, the tool performs adaptive leakage-power optimization by default. The adaptive leakage-power optimization feature uses high-threshold-voltage cells during the initial stages of optimization and gradually increases the use of low-threshold-voltage cells during the later stages.

Setting Up for Dynamic-Power Optimization

To set up for dynamic-power optimization, use the following steps:

1. For a multicorner-multimode design, select the scenarios for dynamic-power optimization by using the `set_scenario_options -dynamic_power true` command.

For more information about selecting the scenarios for dynamic-power optimization, see “[Enabling and Disabling Scenarios for Specific Tasks](#)” on page 3-37.

2. Read the switching activity file by using the `read_saif` command.

For more information about reading in the switching activity file, see “[Annotating the Switching Activity](#)” on page 3-32.

Setting Up for Leakage-Power Optimization

To set up for leakage-power optimization, use the following steps:

1. For a multicorner-multimode design, select scenarios for leakage-power optimization by using the `set_scenario_options -leakage_power true` command.

For more information about selecting scenarios for leakage-power optimization, see [“Enabling and Disabling Scenarios for Specific Tasks” on page 3-37](#).

2. Specify the multiple-threshold-voltage constraint by using the `set_multi_vth_constraint` command.

For more information about specifying the multiple-threshold-voltage constraint, see [“Specifying the Multiple-Threshold-Voltage Constraint” on page 3-32](#).

Performing Low-Power Placement

To perform low-power placement, which reduces dynamic power, use the following steps:

1. Set up the design for dynamic-power optimization, as described in [“Setting Up for Dynamic-Power Optimization”](#).
2. Enable low-power placement by using the `set_optimize_pre_cts_power_options -low_power_placement true` command.
3. Perform placement and optimization by using the `place_opt` command.

Performing Leakage-Power Optimization

To perform leakage-power optimization, use the following steps:

1. Set up the design for leakage-power optimization, as described in [“Setting Up for Leakage-Power Optimization”](#).
2. Perform leakage-power optimization by using one of the following commands:

- `place_opt -power`
- `clock_opt -power`
- `psynopt -power`
- `psynopt -only_power`

During leakage-power optimization, the tool reduces the leakage power only if it does not degrade timing. To specify that a large reduction in leakage power is acceptable at the cost of a very small degradation in timing, set the `icc_preroute_tradeoff_timing_for_power_area` variable to `true`.

If you run the `place_opt` or `clock_opt` command without the `-power` option or the `psynopt` command without the `-power` or `-only_power` option, the tool does not consider leakage power during timing optimization. As a result, the leakage-power QoR can degrade. When you are not performing leakage-power optimization, to consider leakage power during timing optimization, set the `icc_preroute_power_aware_optimization` variable to `true`.

Performing Total-Power Optimization

To perform total-power optimization, which reduces both dynamic and leakage power, use the following steps:

1. Set up the design for dynamic-power optimization, as described in “[Setting Up for Dynamic-Power Optimization](#)”.
2. Set up the design for leakage-power optimization, as described in “[Setting Up for Leakage-Power Optimization](#)”.
3. Enable total-power optimization by using the `-effort` option with the `set_total_power_strategy` command.

The `-effort` option supports the following settings:

- `none` (default)

Disables total-power optimization, which is the default.

Although total-power optimization is disabled with this setting, you can still perform leakage-power optimization as described in “[Performing Leakage-Power Optimization](#)”.

- `medium`

Enables total-power optimization.

- `high`

Enables a more aggressive total-power optimization, which can increase the runtime.

To see the current setting for the total-power strategy, use the `report_total_power_strategy` command.

4. Perform total-power optimization by using one of the following commands:

- `place_opt -power`
- `psynopt -power`
- `psynopt -only_power`

When you perform total-power optimization, the tool ignores the following settings:

- The multiple-threshold-voltage constraint set by using the `set_multi_vth_constraint` command.
- The setting of the `icc_preroute_tradeoff_timing_for_power_area` variable.

Optimizing Clock-Gating Logic

If the clock-enable paths in your design are timing critical, you can perform topology-based restructuring and optimization of the clock-gating logic by using the `place_opt -optimize_icgs` command. Alternatively, you can use the `create_placement -optimize_icgs` command followed by the `place_opt -optimize_icgs -skip_initial_placement` command.

Before you optimize the clock-gating logic,

- Identify the library cells that you want the tool to exclude when optimizing the clock gating logic by using the `set_dont_use -power` command.
- Restrict the optimization of clock gating cells by using
 - The `set_dont_touch` command to identify the cells you do not want to optimize.
 - The `set_size_only` command to identify the cells you want to size only.
- Remove clock latency values specified on clock-gating cells. Applying clock latency values on clock-gating cells is not recommended.
- Specify all the clock-tree-synthesis-related constraints and settings for the design, including the clock tree synthesis scenarios. The `place_opt -optimize_icgs` command requires this information to derive the skew margins for the clock enable paths.

Reducing the Total Negative Slack

By default, the tool works on reducing the total negative slack during optimization. However you can increase the effort used for reducing the total negative slack by using the following command setting:

```
icc_shell> set_optimization_strategy -tns_effort high
```

This setting increases the effort used for reducing the total negative slack, at the expense of runtime, when you run the following commands:

- `place_opt`
- `clock_opt -only_psyn`

- psynopt
- preroute_focal_opt
- route_opt
- focal_opt

Omitting Initial Placement

If you have already run the placer on a design, you can skip the initial placement step of the `place_opt` command. To omit initial placement, you specify the `-skip_initial_placement` option of the `place_opt` command. When using this option, you must ensure that every cell has a legal location and all macros are fixed. Otherwise, the `place_opt` command terminates with an error. This option is disabled by default. Do not use this option with the `-spg` option.

When you use the `-skip_initial_placement` option, the tool skips the initial coarse placement phase and starts with the incremental high-fanout synthesis phase. The tool does not remove existing buffer trees during incremental high-fanout synthesis.

Using Physical Guidance From the Design Compiler Tool

You can use the Synopsys physical guidance information in the Design Compiler tool as a starting point for place and route in the IC Compiler tool. You enable the capability in the IC Compiler tool by specifying the `-spg` option of the `place_opt` command.

Using Design Compiler placement provides the following benefits:

- Reduces the runtime of the placement step
- Achieves better correlation between the IC Compiler and Design Compiler tools

You can transfer the Design Compiler placement to the IC Compiler tool, in either a .ddc file, a Milkyway design, or a DEF file. If you use a DEF file to transfer the placement information, you also need to transfer the design database in the Verilog format, along with constraint files such as SDC and UPF.

Note:

Before using a DEF file to transfer the placement information, you need to set the `spg_enable_ascii_flow` variable to true.

The Design Compiler placement information contains only the placement information for the standard cells and macros cells in the design. It does not contain any floorplan information. Reapply the floorplan and other physical constraints, before you run the `place_opt -spg` command. To preserve placement consistency and improve correlation with the Design

Compiler results, use the same floorplan and physical constraints used in the Design Compiler tool. Do not change the physical constraints, such as the die area, macro placements, port locations, and move bounds. However, you can provide additional floorplan information that is not currently supported by the Design Compiler tool, but is needed in the IC Compiler tool for physical implementation.

The following example shows how to use the Synopsys physical guidance with the `place_opt` command:

```
icc_shell> read_ddc design_spg.ddc
icc_shell> read_def floorplan.def
icc_shell> place_opt -area_recovery -power -congestion -optimize_dft -spg
```

Note:

The Synopsys physical guidance feature is only available in the Design Compiler Graphical tool.

For more information, see the Synopsys physical guidance information in the *Design Compiler User Guide*.

Creating Multibit Register Banks

You can use the register banking flow to merge single-bit registers to form multibit register banks during placement and optimization.

Creating multibit register banks reduces

- The design area due to the smaller area of one multibit register bank compared to that of multiple single-bit registers
- The clock tree power consumption due to the smaller number of clock tree buffers

The register banking flow consists of the following steps:

1. [Preparing for Register Banking](#)
2. [Performing Initial Placement](#)
3. [Performing Register Banking](#)
4. [Performing Optimization](#)
5. (Optional) [Splitting Register Banks](#)

Preparing for Register Banking

To specify the settings for register banking, use the `set_banking_guidance_strategy` command.

When you use the `set_banking_guidance_strategy` command, you must:

- Specify the name of the map file by using the `-input_map_file` option.

The map file defines the library cells to use for register banking.

The following is an example of an input map file:

```
4 {1 MREG4}
6 {1 MREG2}{1 MREG4}
8 {2 MREG4}
10 {1 MREG2}{2 MREG4}
```

In this map file,

- The first line specifies that four single-bit registers can be combined to form a register bank consisting of one MREG4 library cell.
 - The second line specifies that six single-bit registers can be combined to form a register bank consisting of one MREG2 library cell and one MREG4 library cell.
 - The third line specifies that eight single-bit registers can be combined to form a register bank consisting of two MREG4 library cells.
 - The fourth line specifies that ten single-bit registers can be combined to form a register bank consisting of one MREG2 library cell and two MREG4 library cells.
- Specify the name of the generated register banking file by using the `-output_file` option.

The tool generates the register banking file when you run the `create_placement` or `place_opt` command. This file contains a series of `create_register_bank` commands that you can use to perform register banking.

You can also specify the following options for register banking:

- To specify the minimum or maximum number of registers to group together, use the `-minimum_flop_count` or `-maximum_flop_count` option.
- To specify a list of instances to exclude, use the `-exclude_instances` option.
- To specify a list of library cells for which to exclude all instances, use the `-exclude_library_cells` option.
- To specify that registers with `size_only` attributes should be excluded, use the `-exclude_size_only_flops` option.

- To specify that registers that are the first or last register of a scan chain should be excluded, use the `-exclude_start_stop_scan_flops` option.
 - To specify the percentage of critical registers with the worst negative slack to exclude, use the `-wns_threshold` option. By default, the tool includes all registers.
- To specify a different percentage of critical registers for each path group, use the `-wns_threshold_file` option with a threshold file.

The following is an example of a threshold file:

```
GC_CLK : 25
M_CLK : 15
```

Each line in the threshold file contains a name of a path group and the percentage of critical registers to ignore for that path group.

If you specify both the `-wns_threshold` and `-wns_threshold_file` options, the tool ignores the `-wns_threshold` option.

- To specify which types of single-bit library cells can be combined and replaced by which types of multibit library cells, use the `-register_group_file` option with a register group file.

The following is an example of a register group file:

```
reg_group_1 3 {REGX1 REGX2 REGX4} {MREG2 MREG4}
reg_group_2 2 {REGNX1 REGNX2} {MREG2N MREG4N}
```

In this register group file,

- The first line specifies that single-bit registers of the three library cell types REGX1, REGX2, and REGX4 can be combined to form a register bank consisting of library cell types MREG2 or MREG4.
- The second line specifies that single-bit registers of the two library cell types REGNX1 and REGNX2 can be combined to form a register bank consisting of library cell types MREG2N or MREG4N.
- To specify that a specific pin of the single-bit registers must be connected to the same net to be grouped together, use the `-common_net_pins` option.

For example, to specify that the RN pin of the single-bit registers must be connected to the same net for them to be grouped together, use the `-common_net_pins RN` option setting.

- To specify a prefix for the names of the register banks in the register banking file, use the `-name_prefix` option. The tool uses the name prefix you specify to generate a unique

name for the `-name` option of each `create_register_bank` command in the generated register banking file.

The name of each register bank has the *prefixA_B* format, where *prefix* is the name prefix you specify and *A* and *B* are integers. By default, the tool uses *group* as the name prefix and derives a unique name for each register bank starting with the name *group0_0*.

The register banking settings you specify by using the `set_banking_guidance_strategy` command apply only to the current session of the tool. The settings are not saved in the Milkyway design database when you save the design.

Use the `report_banking_guidance_strategy` command to report the register banking settings.

Use the `remove_banking_guidance_strategy` command to remove the register banking settings.

Performing Initial Placement

Perform initial placement by using the `create_placement` command.

During placement, the tool uses the register banking settings you specified and identifies the single-bit registers that can be combined to form multibit register banks. The tool then generates a register banking file, which contains a series of `create_register_bank` commands, that is used to create the register banks.

The tool also generates a register banking file when you run the `place_opt` command. However, the tool does not generate a register banking file when you run the following commands:

- `create_placement -optimize_icgs`
- `create_placement -timing_driven`
- `place_opt -skip_initial_placement`
- `place_opt -optimize_dft`
- `place_opt -optimize_icgs`

The following is an example of a register banking file that the tool generates:

```
create_register_bank -name group0_0 \
{i_REG0 i_REG1 i_REG2 i_REG3} -lib_cell MREG4
create_register_bank -name group0_1 \
{i_REG4 i_REG5} -lib_cell MREG2
```

Performing Register Banking

View the register banking file that the tool generated during initial placement and modify the `create_register_bank` commands if necessary. Perform register banking by sourcing the register banking file.

Performing Optimization

Optimize the design by using the `place_opt -skip_initial_placement` command.

Splitting Register Banks

If necessary, you can split a register bank into smaller register banks or single-bit registers by using the `split_register_bank` command.

The following example shows how to split a register bank named `reg_group_0`, which is a four-bit register bank, into two two-bit register banks:

```
icc_shell> split_register_bank reg_group_0 -lib_cells {MREG2 MREG2}
```

Outputting Additional Information During Preroute Optimization

To output additional information during the different stages of optimization, set the `preroute_opt_verbose` variable before you run the `place_opt`, `psynopt`, `preroute_focal_opt`, or `clock_opt -only_psyn` command.

You specify an eight-bit binary value for the `preroute_opt_verbose` variable. You use the different bits of this binary value to specify the different stages of optimization during which additional information is output, as shown in [Table 4-4](#).

Table 4-4 Bitwise settings for preroute_opt_verbose

To do this	Set this bit
Output additional information during DRC violation fixing.	2
Output additional information during hold violation fixing.	3
Output additional information during tie-off violation fixing.	4
Output additional information during multi-port-net violation fixing.	5
Output additional information during setup violation fixing.	6
Save the information for the setup violation fixing, DRC violation fixing, or both to a file named <code>propt_verbose.log</code> .	8

Note:

When you set bit eight, you must also enable information for the setup violation fixing stage by setting bit six or DRC violation fixing stage by setting bit two.

You can specify the binary value as a decimal or hexadecimal number. For example, to output information when fixing DRC violations, set the `preroute_opt_verbose` variable to 2, which corresponds to binary 0000 0010.

```
icc_shell> set_app_var preroute_opt_verbose 2
```

You can output information for more than one optimization stage by setting multiple bits. For example, to output information when fixing DRC, hold, and setup violations, set the `preroute_opt_verbose` variable to 38, which corresponds to binary 0010 0110.

```
icc_shell> set_app_var preroute_opt_verbose 38
```

Saving Intermediate Results During Preroute Optimization

You can enable `place_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 `place_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, such as `report_timing`, `report_qor`, `extract_rc`, `legalize_placement`, `create_scenario`, `current_scenario`, `all_scenarios`, `set_active_scenarios`, `remove_scenario`, and `route_zrt_global`. You can also use the commands on block abstractions created from a checkpoint design for top-level analysis.

The following commands are not allowed on a checkpoint design or block abstraction created from a checkpoint design: `place_opt`, `psynopt`, `create_buffer_tree`, `clock_opt`, `compile_clock_tree`, `route_opt`, `skew_opt`, `optimize_clock_tree`, `optimize_dft`, `optimize_fp_timing`, and `optimize_pre_cts_power`. Note that IC Compiler terminates any of these commands with an error if you try to use them on a checkpoint design.

Note:

This functionality does not apply to block abstractions.

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

```
icc_shell> set_checkpoint_strategy -enable -prefix "runx" -overwrite  
icc_shell> place_opt
```

The resulting checkpoint design is

```
runx_place_opt_single_checkpoint_MYDESIGN_080908_120402_1
```

Each subsequent run with the `-overwrite` option overwrites the previous checkpoint design and uses the same checkpoint design name.

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 `place_opt` only, enter

```
icc_shell> remove_checkpoint_designs -command place_opt
```

To disable checkpointing, enter

```
icc_shell> set_checkpoint_strategy -disable
```

Analyzing Placement and Optimization Results

The following topics describe features that you can use to analyze your placement and optimization results:

- [Analyzing Placement and Optimization Feasibility](#)
- [Analyzing Cell Displacement During Legalization](#)
- [Reporting Cells Added During Optimization](#)
- [Analyzing Cells and Nets With `dont_touch` Attributes](#)
- [Reporting Unfixed Violations](#)
- [Generating Physical Data Reports](#)
- [Analyzing the Placement Area Utilization](#)
- [Analyzing the Timing Results](#)
- [Analyzing Power](#)
- [Reporting Quality-of-Results](#)
- [Analyzing Timing Results in the GUI](#)

- [Analyzing the Placement in the GUI](#)
- [Viewing Scan Chains](#)

Analyzing Placement and Optimization Feasibility

During placement and optimization, you might need to run the `place_opt` command multiple times to obtain optimal results. To improve runtime and to reduce iterations of placement and optimization during early design stages, you can use the `place_opt_feasibility` command to analyze placement and optimization feasibility. Running the feasibility flow can minimize the number of iterations, validate timing constraints, and report all paths that are not feasible.

[Table 4-5](#) defines the `place_opt_feasibility` options.

Table 4-5 The place_opt_feasibility Options

To do this	Use this option
Enable congestion removal to improve routability.	<code>-congestion</code>
Specify to which stage the command should run.	<code>-stage initial_placement drc first_opt</code>
Specify which stage the command should skip.	<code>-skip initial_placement drc first_opt</code>
Enable scan chain reordering if scan chains exist.	<code>-optimize_dft</code>
Enable leakage-power optimization.	<code>-power</code>
Recover area for cells that are not on the critical timing paths.	<code>-area_recovery</code>
Use the zero wire load model to analyze timing.	<code>-zero_wire_load</code>
Specify to consider the scan chain connections during placement. By default, the command ignores scan nets.	<code>-consider_scan</code>
Specify to continue placement when the design contains scan chains but no SCANDEF data. By default, missing SCANDEF data causes the command to exit with an error message. Setting this option enables the placer to continue with a warning and results in reduced QoR.	<code>-continue_on_missing_scandef</code>

The feasibility flow performs the following tasks:

- Performs quick initial placement, high-fanout net synthesis, logical DRC checking, and timing optimization.
- Invokes the `check_physical_design` and `check_scenarios` commands internally to check the design and libraries.
- Checks and reports timing endpoints that are not feasible and bottlenecks in timing constraints or floorplans.

If your design is congested, you should specify the `-congestion` option to achieve better correlation with the `place_opt` flow.

Using the feasibility flow results in faster placement and optimization relative to the `place_opt` flow.

Analyzing Cell Displacement During Legalization

Floorplans that contain narrow channels, high local congestion, and poor placement of blockages or macros can lead to large cell displacement during legalization. Large cell displacement can affect timing optimization and degrade the quality of results. To identify the cause of potential large cell displacement in your designs, you can perform cell displacement analysis after running placement and optimization.

Cell displacement analysis performs the following tasks:

- Correlates large cell displacement with critical timing paths and congestion maps.
- Reports displacement count that occurs during each legalization stage.
- Plots displacement vectors for each legalization stage if the appropriate layout window is open.

A file named `feas_cell_history`, which contains the cell location history, is automatically generated by the tool when you run the `place_opt_feasibility` or `clock_opt_feasibility` command. To perform cell displacement analysis, use the

`analyze_displacement` command with the mandatory `-input` option to provide the cell location history file, as shown in the following example:

```
icc_shell> analyze_displacement -input feas_cell_history
...
*****
Minimal cell row height: 5.040.

Summary of analysis Result
-----
Checkpoint 1: Start_legalizer
    39 new large displacement to next checkpoint
Checkpoint 2: End_legalizer
    0 new large displacement to next checkpoint
Checkpoint 3: Start_legalizer
    42 new large displacement to next checkpoint
Checkpoint 4: End_legalizer
    0 new large displacement to next checkpoint
Checkpoint 5: Current.
1
```

You can generate a cell location history file for other placement and optimization commands by using the `set_displacement_analysis_options -enable` command. The following example shows how you can track and analyze the cell displacement that occurs during legalization when you run the `place_opt` command:

```
icc_shell> set_displacement_analysis_options -enable \
    -output place_opt.cell_disp
icc_shell> place_opt
icc_shell> analyze_displacement -displacement_threshold 0 \
    -input place_opt.cell_disp
```

You can also track the cell displacement that occurs when you manually move cells by using the `record_cell_locations` command with the `set_displacement_analysis_options -enable` command, as shown in the following example:

```
icc_shell> set_displacement_analysis_options -enable \
    -output move_obj.cell_disp
icc_shell> record_cell_locations -checkpoint before_move
icc_shell> move_objects -delta {100 150} [get_selection]
icc_shell> record_cell_locations -checkpoint after_move
icc_shell> analyze_displacement -displacement_threshold 0 \
    -input move_obj.cell_disp
```

You can report the options specified for the `set_displacement_analysis_options` command by using the `report_displacement_analysis_options` command.

To identify the timing impact of large cell displacement, you can specify both the `-move_back` and `-compare_timing` options with the `analyze_displacement` command. Use the `-move_back` option to reverse the displacement of cells to the locations of the specified stage, and use the `-compare_timing` option to compare the timing between the specified stage and the final placement. The `-compare_timing` option performs preroute

RC estimation, updates timing, and regenerates the timing report. You can specify which timing paths to compare by using the `-timing_path` option. To preserve the results of the `-move_back` option, you should use the `-commit_change` option.

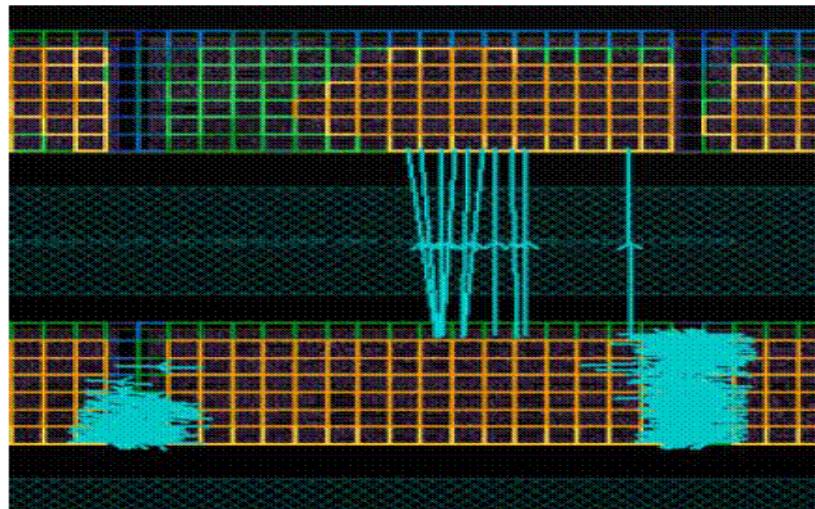
For example, the following command moves the displaced cells back to the locations of legalization stage 3 and reports timing comparison information for the timing paths in the `report.tim` file.

```
icc_shell> analyze_displacement -input feas_cell_history \
    -move_back 3 -compare_timing -timing_path report.tim
```

To find out if specific cells have large cell displacement, use the `-cells` option.

You can also view cell displacement by choosing Placement > Large Displacement Analysis in the GUI. [Figure 4-7](#) displays the large cell displacement as an overlay over the cell density map.

Figure 4-7 Cell Density Map With Large Cell Displacement



Reporting Cells Added During Optimization

Use the `report_optimization_created_cells` command to report the cells added to the design when you run the `place_opt`, `psynopt`, `clock_opt`, `route_opt`, `focal_opt`, `preroute_focal_opt`, `place_opt_feasibility`, or `clock_opt_feasibility` command.

By default, the `report_optimization_created_cells` command generates a summary of all the cells added during the different stages of optimization, such as setup fixing, DRC fixing, hold fixing, high-fanout-net synthesis, tie-cell optimization, and multiple-port-net fixing. You can report the cells created during a specific stage by using the

`-optimization_stage` option. The valid values for this option are `setup`, `drc`, `hold`, `hfs`, `tie_optimization`, and `port_fixing`.

If you run the `report_optimization_created_cells` command after running several optimization commands, the tool reports the cells added by all the commands. To help identify the cells that are added by a specific optimization command, set the `compile_instance_name_prefix` variable to a unique string before you run that command, as shown in the following example:

```
icc_shell> set_app_var compile_instance_name_prefix place_opt_cell_
icc_shell> place_opt
icc_shell> set_app_var compile_instance_name_prefix clock_opt_cell_
icc_shell> clock_opt
icc_shell> report_optimization_created_cells -optimization_stage setup
```

The following is a portion of the report generated by the `report_optimization_created_cells` command in the previous example:

```
*****
Report : optimization created cells
Design : design1
Version: I-2013.12-ICC
Date   : Fri Oct 25 12:51:49 2013
*****
```

Optimization Stage Descriptions:

<code>setup</code>	- cell created at setup optimization stage
--------------------	--

Cell	Optimization Stage
<code>clock_opt_cell_251</code>	<code>setup</code>
<code>clock_opt_cell_250</code>	<code>setup</code>
<code>clock_opt_cell_249</code>	<code>setup</code>
<code>clock_opt_cell_248</code>	<code>setup</code>
...	
...	
<code>place_opt_cell_876</code>	<code>setup</code>
<code>place_opt_cell_875</code>	<code>setup</code>
<code>place_opt_cell_874</code>	<code>setup</code>
<code>place_opt_cell_873</code>	<code>setup</code>
...	
...	

You can query the `optimization_stage` cell attribute to create a collection of cells added during a specific optimization stage. The valid values for this attribute are `setup`, `drc`, `hold`, `hfs`, `tie_optimization`, and `port_fixing`. The following example creates a collection of cells added during setup fixing:

```
icc_shell> get_cells -hierarchical -filter "optimization_stage==setup"
```

Analyzing Cells and Nets With `dont_touch` Attributes

The IC Compiler tool does not optimize cells or nets with `dont_touch` attributes. Identifying and removing the `dont_touch` attributes from cells and nets, if appropriate, can improve QoR.

To report all the don't touch cells and nets in the design and the type of don't touch that applies to each reported object, use the `report_dont_touch` command. In the report, the table legend provides a description of the types of don't touch settings that are applicable for the reported objects.

The following example shows a report of all the don't touch cells in a design:

```
icc_shell> report_dont_touch -class cell -nosplit
...
Description for Cell dont_touch Types:
  inh_ref - Cell inherits dont_touch from reference design or library cell
  ph_fixed_placement - Cell with fixed placement

Object          Class      Types
-----
u_ram1         cell      ph_fixed_placement
u_ram2         cell      ph_fixed_placement
u_xt/u_8/u_mem cell      inh_ref
-----
Total 3 dont_touch cells
1
```

The following example shows a don't touch report for a specific net:

```
icc_shell> report_dont_touch [get_nets Multiplier/Product[0]]
...
Description for Net dont_touch Types:
  inh_parent - Net inherits dont_touch from parent
  mv_iso - Prevents buffering between isolation cells and power domain boundary

Object          Class      Types
-----
Multiplier/Product[0]    net      inh_parent, mv_iso
-----
```

In this example, the `Multiplier/Product[0]` net is a hierarchical net that is inside a block with an explicit don't touch setting. As a result, it inherits this setting from its parent. The net also has an implicit don't touch setting as a result of multivoltage synthesis.

To get an alphabetically sorted list of cell and net don't touch types applicable for your design and their descriptions, use the `list_dont_touch_types` command. This command is useful when you want to look up the don't touch types applicable for your design for building a collection of don't touch objects.

To create a collection of don't touch cells and nets in the current design, use the `get_dont_touch_cells` and `get_dont_touch_nets` commands, respectively. You can use these commands at the command prompt, nest them as an argument to another command, or assign the command results to a variable.

When issued from the command prompt, by default, the `get_dont_touch_cells` and `get_dont_touch_nets` commands display a maximum of 100 objects. You can change this value by setting the `collection_result_display_limit` variable.

The following example shows a query for don't touch cells that result from an ideal network setting:

```
icc_shell> get_dont_touch_cells -type idn InstDecode/*
{InstDecode/reset_UPF_LS InstDecode/U38 InstDecode/U36}
```

The following example shows a query for don't touch nets that result from multivoltage settings:

```
icc_shell> get_dont_touch_nets -type mv* block1/NET*
{block1/NET1QNX block1/NET2QNX}
```

Reporting Unfixed Violations

To generate an unfixed violations report, run the `analyze_design_violations -stage preroute` command. The tool can analyze a single violation type per run of the `analyze_design_violations` command, which must be specified with the `-type` option:

- To report information about the hold time violations, use the `-type hold` option with the `analyze_design_violations` command.

By default, the report includes information about all hold time violations. To restrict the results to information about specific endpoints, use the `-input` option to specify a file that contains the endpoints from one of the following commands (depending on whether your design is a multicorner-multimode design):

```
icc_shell> report_constraint -all_violators -min_delay -nosplit \
-significant_digits 5

icc_shell> report_constraint -all_violators -min_delay -nosplit \
-significant_digits 5 -scenarios [all_active_scenarios]
```

The endpoint lines should look similar to the following:

I_STACK_TOP/n342	0.70	0.89	-0.19	(VIOLATED)
------------------	------	------	-------	------------

- To report information about the maximum transition time violations, use the `-type max_trans` option with the `analyze_design_violations` command.

By default, the report includes information about all maximum transition time violations. To restrict the results to information about specific nets, use the `-input` option to specify

a file that contains the nets from one of the following commands (depending on whether your design is a multicorner-multimode design):

```
icc_shell> report_constraint -all_violators -max_transition -nosplit \
-sigrificat_digits 5

icc_shell> report_constraint -all_violators -max_transition -nosplit \
-sigrificat_digits 5 -scenarios [all_active_scenarios]
```

You must include both the violated net and the affected pins for each net you want to analyze. For example,

Net	Required	Actual	Slack
	Transition	Transition	
n234	2.00	2.01	-0.01 (VIOLATED)
PIN : U52/Z	2.00	2.01	-0.01 (VIOLATED)

You can control the thresholds used by this command for high-fanout nets and small violations. By default, the tool considers nets with more than 30 fanouts as high-fanout nets and violations of less than 5 ps as small violations. To change the high-fanout net threshold, use the `-fanout` option. To change the small violation threshold, use the `-slack` option.

By default, the tool writes the results to a directory named `analyze_design_violations`. To change the output directory, use the `-output` option.

Generating Physical Data Reports

You can run statistical analysis on the design to perform a quick check or to analyze the optimization results before or after an optimization stage, including `place_opt`, `clock_opt`, and `route_opt`. To perform statistical analysis, specify the `-design_statistics` option with the `check_physical_design` command. The statistical analysis reports the design statistics, such as cells, nets, DRC violations, and area, but not floorplan limitations or timing constraints. For example,

```
icc_shell> check_physical_design -design_statistics
```

To include a clock tree report with the statistical analysis, use the `-include_clock_tree_report` option with the `-design_statistics` option. For example,

```
icc_shell> check_physical_design -design_statistics \
    -include_clock_tree_report
```

The example creates a `cpd_design_statistics_datetime` directory that stores the design statistics report named `index.html` and the statistical analysis tables in HTML format. The `index.html` file summarizes the design statistics in tabular format and provides a link to each HTML table. You click the More Info link under each category to view the HTML table for

detailed statistics. To view the design statistics report, use a Web browser, such as Firefox, as shown in the following example:

```
% firefox cpd_design_statistics_datetime/index.html
```

[Figure 4-8](#) shows an example of the design statistics report.

Figure 4-8 Design Statistics Report

DESIGN STATISTICS REPORT

Short Summary for Design : ORCA			
Date : Thu Jul 28 12:26:57 2011			
1)	Number of Std cells in design :	22209	More Info
2)	Number of Inv/Buf cells in design :	6457	More Info
3)	Number of ICG cells in design :	0	
4)	Number of Macro cells in design :	13	More Info
5)	Number of Physical Only cells in design :	20	More Info
6)	Number of dont_touch cells in design :	0	0 % of Std
7)	Number of size_only cells in design :	24	0 % of Std
8)	Number of fixed_placement cells in design :	0	0 % of Std
9)	Number of Buffers Avail in libraries :	26	More Info
10)	Number of Buffers with dont_use :	0	
11)	Number of Inverters Avail in libraries :	13	More Info
12)	Number of Inverters with dont_use :	0	
13)	Number of Sequential cells in design :	2959	More Info

To direct the output of the `check_physical_design` command to a specified directory, use the `-output` option. To direct the output of the command to a text file, use the UNIX redirection operators (`>` or `>>`) or the `redirect` command. For example,

```
icc_shell> check_physical_design -design_statistics > my_file
```

You cannot use the `-design_statistics` option with the `-stage` option.

Analyzing the Placement Area Utilization

Placement area utilization, or simply “utilization,” means the percentage of area available for placement that is already occupied by placed cells. For example, a utilization of 80 percent means that 80 percent of the available area is occupied by cells and 20 percent is empty and can still be used for additional cell placement, for movement of cells for legalization and optimization, or as an allowance to prevent excessive routing congestion.

You can report the utilization for part or all of the design with the `report_placement_utilization` command.

This syntax of this command is

```
report_placement_utilization
  [-non_fixed_only]
  [-grid_size float]
  [-verbose]
  [-coordinates {llx lly urx ury}]
```

For example, to report the utilization of the rectangular area bounded by the corner points (340, 380) and (400, 450), you would use the following command:

```
icc_shell> report_placement_utilization -coordinates {340 380 400 450}
...
Std cell utilization:      84.02%  (2193/(2610-0),
                           (std_cell_area/(region_area-blockage_area))
                           (Non-fixed + Fixed)
Std cell utilization:      77.66%  (1450/(2610-743))
                           (Non-fixed only)
Total region area:        2610    sites, bbox
                           (340.00 380.00 400.00 450.00) um
Std cell area :            2193    sites, (non-fixed:1450, fixed:743)
Macro cell area :          0       sites
Blkgs exclude fixed cells: 0       sites
Blkgs include fixed cells: 743    sites
Std cell count :           301    (non-fixed:183, fixed:118)
Macro cell count :          0       sites
Pnet area :                0       sites

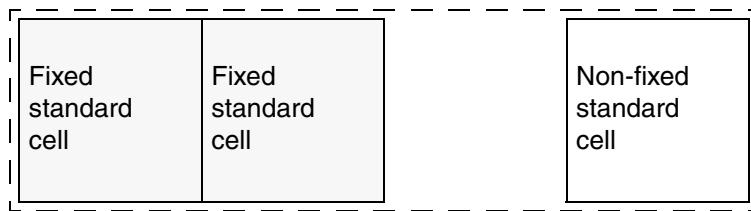
*****
Sub-Region Utilization
*****
Number of regions with placement utilization 0 - 0.125 is 0 (0.00%)
Number of regions with placement utilization 0.125 - 0.25 is 0 (0.00%)
Number of regions with placement utilization 0.25 - 0.375 is 0 (0.00%)
Number of regions with placement utilization 0.375 - 0.5 is 0 (0.00%)
Number of regions with placement utilization 0.5 - 0.625 is 0 (0.00%)
Number of regions with placement utilization 0.625 - 0.75 is 0 (0.00%)
Number of regions with placement utilization 0.75 - 0.875 is 0 (0.00%)
Number of regions with placement utilization 0.875 - 1 is 16 (100.00%)
```

If you do not specify the coordinates, the command reports the utilization of the whole chip.

In the reporting of utilization, there are two basic ways to consider the presence of standard cells that have fixed placement (“fixed standard cells”). These cells can be considered the same as unfixed cells, in other words, as cells occupying the available space. Conversely, the space occupied by fixed cells can be considered unavailable space, in what is called a “non-fixed-only” utilization report.

Consider the example shown in [Figure 4-9](#). This area of four square units is occupied by three cells, each with an area of one square unit, leaving one square unit of empty space.

Figure 4-9 Non-Fixed-Only Utilization Example



Default utilization (fixed and unfixed standard cells): $3/4 = 75$ percent
Non-fixed-only standard cell utilization: $1/2 = 50$ percent

The default utilization is reported to be 75 percent because $3/4$ of the available area is occupied by standard cells. However, for non-fixed-only utilization reporting, the area occupied by fixed standard cells is considered unavailable, so the available area is two square units, occupied by one square unit of non-fixed standard cells, for a utilization of $1/2$ or 50 percent.

Utilization reporting also considers the presence of placement blockages. For both default and non-fixed-only reporting, any part of the area occupied by a blockage is considered unavailable for placement. Thus, the default utilization is calculated as follows:

$(\text{non-fixed_standard_cell_area} + \text{fixed_standard_cell_area}) / (\text{total_area} - \text{blocked_area})$

whereas non-fixed-only utilization is calculated as follows:

$(\text{non-fixed_standard_cell_area}) / (\text{total_area} - \text{fixed_standard_cell_area} - \text{blocked_area})$

The `report_placement_utilization` command reports both the default and non-fixed-only utilization values. It also divides the specified region into subregions and separately reports the utilization within those subregions, to give an idea of the “evenness” of the utilization. By default, the subregion size is five times the placement site height. You can specify a different subregion size, in microns, by using the `-grid_size` option.

The `-verbose` option causes the reporting of the dimensions, locations, and individual utilization values of the subregions. Otherwise, the report shows only a summary listing of the number and percentage of subregions having different ranges of utilization values. The `-non_fixed_only` option specifies whether to use default or non-fixed-only utilization values for the subregion utilization report.

Analyzing the Timing Results

After you set the timing constraints such as clocks, input delays, and output delays, it is a good idea to use the `check_timing` command to check for timing setup problems and timing conditions such as incorrectly specified generated clocks and combinational feedback loops. The command checks the timing attributes of the current design and issues warning messages about any unusual conditions found.

This is the full syntax of the `check_timing` command:

```
check_timing
[-overlap_tolerance minimum_distance]
[-override_defaults check_list]
[-include check_list]
[-exclude check_list]
[-multiple_clock]
[-retain]
```

These are the types of timing checks that can be performed:

```
clock_crossing
data_check_multiple_clock
data_check_no_clock
generated_clocks
generic
loops
multiple_clock
no_input_delay
retain
unconstrained_endpoints
pulse_clock_cell_type
gated_clock
ideal_timing
clock_no_period
```

The types of timing checks shown in boldface are performed by default. You can include or exclude specific types of timing checks by using the `-include` and `-exclude` options or by setting the `timing_check_defaults` variable.

After placement, you can use the `report_timing` command to report the worst-case timing paths in the design. By default, the command reports the path having the worst maximum (setup) delay in each clock group. Net delays are based on estimated route lengths.

For example, here is a typical `report_timing` command:

```
icc_shell> report_timing
...
Startpoint: I_SDRAM_TOP/I_SDRAM_IF/DQ_out_1_reg_8_
(falling edge-triggered flip-flop clocked by SDRAM_CLK)
Endpoint: sd_DQ_out[8]
(output port clocked by SD_DDR_CLK)
Path Group: SD_DDR_CLK
Path Type: max

Point                                     Incr      Path
-----
clock SDRAM_CLK (fall edge)                3.75     3.75
clock network delay (ideal)                0.00     3.75
I_SDRAM_TOP/I_SDRAM_IF/DQ_out_1_reg_8_/CPN (sdcfq1) 0.00     3.75 f
I_SDRAM_TOP/I_SDRAM_IF/DQ_out_1_reg_8_/Q (sdcfq1)   0.36     4.11 r
I_SDRAM_TOP/I_SDRAM_IF/sd_mux_dq_out_8/Z (mx02d4)  0.21 *    4.31 r
I_SDRAM_TOP/I_SDRAM_IF/sd_DQ_out[8] (SDRAM_IF)      0.00     4.31 r
I_SDRAM_TOP/sd_DQ_out[8] (SDRAM_TOP)          0.00     4.31 r
sd_DQ_out[8] (out)                         0.00     4.31 r
data arrival time                          0.00     4.31

clock SD_DDR_CLK (rise edge)               7.50     7.50
clock network delay (ideal)                1.00     8.50
clock uncertainty                         -0.05    8.45
output external delay                     -2.00    6.45
data required time                        0.00     6.45
-----
data required time                         6.45
data arrival time                         -4.31
-----
slack (MET)                                2.14
```

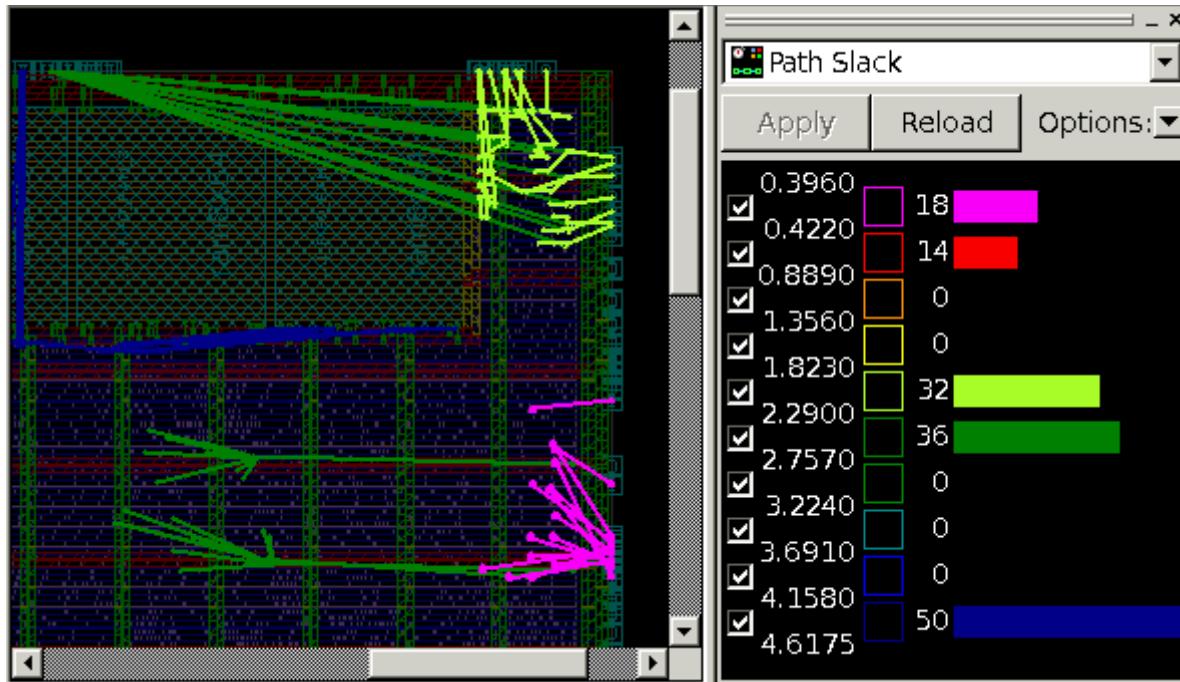
The default report shows the startpoint, endpoint, path group (clock domain), path type (minimum delay, maximum delay, max_rise, min_fall, and so on), the incremental and cumulative time delay values along the data and clock paths, the data required time at the path endpoint, and the timing slack for the path.

The `report_timing` command options let you specify the scope of paths reported (from, to, or through specified points in the design), the path types reported, the numbers of worst-case paths reported, and the types of information reported for intermediate points in the path (transition times, capacitance, net delays, and so on).

Using the GUI, you can generate color-coded diagrams showing the locations of worst-case timing conditions such as path slack, cell slack, net capacitance, clock latency/transition, and crosstalk. From the Visual Mode panel (View > Visual Mode) select the desired type of visual analysis, and then click the Reload button and OK button. An example of a path slack display is shown in [Figure 4-10](#). The layout view shows the worst-case timing paths with the pins and flylines of the paths color-coded according to amount of slack. For more information

about generating these views, see the “Visual Mode panel” topic in IC Compiler Help, or see [Appendix A, “Using GUI Tools.”](#)

Figure 4-10 Path Slack Visual Mode



To get a detailed report on the delay calculation at a given point along a timing path, use the `report_delay_calculation` command. This is the command syntax:

```
report_delay_calculation
  -from from_pin -to to_pin
  [-min] [-max]
  [-nosplit]
  [-crosstalk]
  [-from_rise_transition value]
  [-from_fall_transition value]
```

Specify the “from” and “to” pins of the cell or net that you want to report. These two pins can be the input and output pins of a cell to report the calculation of a cell delay, or can be the driver pin and a load pin of a net to report the calculation of a net delay.

The following command reports in detail the delay calculation between two pins of a cell:

```
icc_shell> report_delay_calculation \
    -from I_SDRAM_TOP/I_SDRAM_IF/DQ_out_1_reg_8_/CPN \
    -to I_SDRAM_TOP/I_SDRAM_IF/DQ_out_1_reg_8_/Q
...
From pin:           I_SDRAM_TOP/I_SDRAM_IF/DQ_out_1_reg_8_/CPN
To pin:            I_SDRAM_TOP/I_SDRAM_IF/DQ_out_1_reg_8_/Q
Main Library Units: 1ns 1pF 1kOhm

Operating Conditions: cb13fs120_ts_max   Library: cb13fs120_ts_max
Library: 'cb13fs120_ts_max'
Library Units: 1ns 1pF 1kOhm
Library Cell: 'sdcfql1'

arc sense:          falling_edge
arc type:           cell
Clock rise transition: 0
Clock fall transition: 0

Rise Delay

cell delay = 0.35504
Table is indexed by
  (X) input_pin_transition = 0
  (Y) output_net_total_cap = 0.0124603
Relevant portion of lookup table:
  (X) 0.0150      (X) 0.2500
(Y) 0.0070      (Z) 0.3380      (Z) 0.3880
(Y) 0.0140      (Z) 0.3640      (Z) 0.4150

Z = A + B*X + C*Y + D*X*Y
A = 0.3089          B = 0.2085
C = 3.7052          D = 0.6079

Z = 0.35504
scaling result for operating conditions
multiplying by 1 gives 0.35504
...
Fall Delay

...
```

Analyzing Power

The `report_power` command calculates and reports power for a design. The command uses the user-annotated switching activity to calculate the net switching power, cell internal power, and cell leakage power, and it displays the calculated values in a power report.

By default, the `report_power` command shows the total dynamic power and leakage power for the whole chip, as shown in the following example:

```
icc_shell> report_power
...
Library(s) Used:

cb13fs120_ts_max (File: /remote/techp5/ref/db/sc_max.db)
ram8x64_max (File: /remote/techp5/ref/db/ram8x64_max.db)
ram16x128_max (File: /remote/techp5/ref/db/ram16x128_max.db)

Operating Conditions: cb13fs120_ts_max Library: cb13fs120_ts_max
Wire Load Model Mode: enclosed
Design           Wire Load Model           Library
-----
ORCA_TOP          ForQA                  cb13fs120_ts_max
PCI_FIFO_1_DW01_sub_1 ForQA                  cb13fs120_ts_max

Global Operating Voltage = 1.08
Power-specific unit information :
  Voltage Units = 1V
  ...
  Cell Internal Power   =  34.8033 mW    (93%)
  Net Switching Power   =   2.7620 mW    (7%)
  -----
  Total Dynamic Power   =  37.5653 mW    (100%)
  Cell Leakage Power    = 424.5199 uW
  ...
  
```

The options of the `report_power` command lets you restrict the scope of the power report to specified instances, to nets or cells alone, or to particular cells or nets obtained by filtering. You can also control the format of the generated report, including the levels of hierarchy reported, the sorting of nets and cells reported, and an optional histogram display of net power.

The following example reports the total power of all cells, including the leakage power of physical-only cells:

```
icc_shell> report_power -cell -only [get_cells -all]
...
Attributes
-----
h - Hierarchical cell

Cell          Cell      Driven Net   Tot Dynamic   Cell
Internal     Internal   Switching Power   (%Cell/Tot) Leakage
Power        Power     Power       Power
-----
```

Cell	Internal Power	Driven Net Power	Tot Dynamic Power	Cell Leakage Power	Attrs
CTS_eclk_delay3	0.0121	0.0162	2.83e-02 (43%)	1.3527	
CKBD16HVT_G2B3I1	9.439e-03	0.0259	3.53e-02 (27%)	1.0109	
CKBD12HVT_G2B3I2	9.384e-03	0.0207	3.01e-02 (31%)	1.0109	
SNPS_dcap_324	0.0000	0.0000	0.0000 (N/A)	0.8000	
...					
Totals (2026cells)	947.939uW	630.335uW	1.573mW (60%)	856.898nW	

The SNPS_dcap_324 cell denotes leakage power consumption of the physical-only cells.

Reporting Quality-of-Results

You can generate a report on the quality of results (QoR) for the design in its current state by using the `create_qor_snapshot` command (or by choosing Timing > Create QoR Snapshot in the GUI). This command measures and reports the quality of the design in terms of timing, design rules, area, power, congestion, clock tree synthesis, routing, and so on. It stores the quality information into a set of snapshot files. You can later retrieve and report the snapshot with the `report_qor_snapshot` command. You can also selectively retrieve, sort, and display the desired information from the snapshot with the `query_qor_snapshot` command (or by choosing Timing > Query QoR Snapshot in the GUI).

`create_qor_snapshot`

You can use the `create_qor_snapshot` command to measure the quality of the design in its current state and store the quality information into a set of report files. You can capture the quality information using different optimization strategies or at different stages of the design and compare the quality results.

Alternatively, you can create a QoR snapshot in the GUI by choosing Timing > Create QoR Snapshot.

The command options let you specify which aspects of quality are to be measured (power, clock tree, and routes) and the conditions for analysis (zero wire load, maximum paths per timing group, and maximum paths per endpoint).

When you use this command, you must at least specify a name for the snapshot by using the `-name` option. For example,

```
icc_shell> create_qor_snapshot -name my_snapshot1
```

The report is written to a set of files into a directory called “snapshot” in the current working directory. After the report is written, you can retrieve the report any time with the `report_qor_snapshot` command, as shown in [Example 4-2](#).

Example 4-2 Reporting a QoR Snapshot

```
icc_shell> report_qor_snapshot
*****
Report          : create_qor_snapshot (my_snapshot1)
Design          : system_controller
Version         : D-2010.03
Date            : Thu Feb 11 13:56:48 2010
Time unit       : 1.0e-09 Second(ns)
Capacitance unit: 1.0e-12 Farad(pF)
Voltage unit    : 1 Volt
Power unit      : N/A
Location        : /remote/PnR/implementation/snapshot
*****
No. of scenario = 1
-----
WNS of each timing group:
-----
Setup WNS:           0.000
Setup TNS:           0.0
Number of setup violations: 0
Hold WNS:            0.000
Hold TNS:            0.000
Number of hold violations: 0
Number of max trans violations: 65
Number of max cap violations: 22
Route drc violations: 0
-----
Area:               451495
Cell count:         23139
Buf/inv cell count: 3088
Std cell utilization: 18.324%
CPU/ELAPSE(hr):    0.11/23.88
Mem(Mb):            1274
Host name:          igcae007
-----
Histogram:
-----
Max violations: ...
```

If your design contains multiple scenarios, you can use the `-scenarios` option to create a snapshot of the specified scenarios. If you do not specify this option, the command creates a snapshot of all active scenarios.

For example, the following command creates a snapshot of the sn1, sn2, and sn3 scenarios:

```
icc_shell> create_qor_snapshot -name test -scenarios {sn1 sn2 sn3}
```

To create a snapshot for the sn1 scenario only, enter

```
icc_shell> create_qor_snapshot -name test -scenarios sn1
```

You can also create a snapshot for the analysis of clock tree synthesis by using the `-clock_tree` option. For example,

```
icc_shell> create_qor_snapshot -name cts_snapshot -clock_tree
```

query_qor_snapshot

The `query_qor_snapshot` command reads in a QoR report generated by previous usage of the `create_qor_snapshot` command and analyzes the results by collecting information about each path. You can query the collected information with various searches and display the results in text or HTML format, or in an interactive HTML window linked to the GUI layout view. The interactive HTML report lets you quickly find paths with certain problems, such as large fanouts or transition degradation, and then view the path in the current design.

Alternatively, you can analyze a QoR snapshot in the GUI by choosing Timing > Query QoR Snapshot.

The command options let you filter and sort the query results and display those results in text or HTML format. The command does not perform any additional analysis of the design, but merely processes the report information already saved in the snapshot report files. Therefore, execution of the `query_qor_snapshot` command is much faster than execution of the `create_qor_snapshot` command.

The following examples demonstrate usage of the `query_qor_snapshot` command.

```
icc_shell> query_qor_snapshot -name preroute -filters "-wns -4.0,-3.0"
...
  Path Group      Start Point      End Point      WNS
  ...            ...
  ...            ...              ...          -3.312
  ...            ...              ...          -3.277
```

This example analyzes the snapshot named preroute and reports the paths having a worst negative slack between -4.0 and -3.0 time units.

```
icc_shell> query_qor_snapshot -name placeopt \
    -filters "-wns ,-1.0 -fanout 40"
...
  Path Group      Start Point      End Point      WNS      Large Fanout
  ...            ...           ...       -3.312     42
  ...            ...           ...       -3.277     42
```

This example reports the paths having a slack worse than -1.0 time units and a fanout greater than 40.

```
icc_shell> query_qor_snapshot -hierarchy
```

Sometimes the paths with the worst timing start in one hierarchical block and end in another. Identifying and grouping these paths can be a challenging task. The `-hierarchy` option automates this process of finding potential path groupings that cross hierarchical boundaries so that the problem paths can be analyzed more efficiently. The `-hierarchy` option, when used by itself, finds all the timing violations that cross block abstractions and hard macros in the design.

The following example finds violating paths that pass through modules A, B, and C:

```
icc_shell> query_qor_snapshot -hierarchy -through {A B C}
```

The following example uses the `-incremental` option, which keeps the previous analysis results (through modules A, B, and C), and applies an additional query about paths starting from module A or B and ending at module A or B.

```
icc_shell> query_qor_snapshot -hierarchy -incremental \
    -from {A/* B/*} -to {A/* B/*}
```

The following example reads the snapshot named max_logic_level and specifies the `-filters` option to report all paths between logic levels 2 and 14.

```
icc_shell> query_qor_snapshot -name max_logic_level \
    -filters "-logic_level 2,15"
```

Note that the paths of logic level 2 are included but the paths of logic level 15 are excluded in the report.

The following example reports paths that start from module C and end at module B, having a worst negative slack worse than -3.0 ns or a fanout more than 40.

```
icc_shell> query_qor_snapshot -name my_snapshot \
    -hierarchy -from top/A/C/* -to top/B/* \
    -filters "-wns ,-3.0 -fanout 40"
```

Analyzing Timing Results in the GUI

In addition to generating timing reports in HTML or text format by using the `query_qor_snapshot` command, you can also use the command to invoke a Categorized Timing Report Interactive Viewer in the GUI. The interactive viewer provides a fast and user-friendly way to analyze timing results during early design stages. To help you debug the causes of timing issues, the viewer contains hypertext links of paths and data that point to the timing reports. To enable you to visualize the selected paths in the design, the viewer cross-highlights the paths in an open layout view.

To open a Categorized Timing Report Interactive Viewer, you must first generate a QoR snapshot by using the `create_qor_snapshot` command and then specify the `query_qor_snapshot -display` command.

For example, to open the viewer with the default filter settings, enter

```
icc_shell> create_qor_snapshot -name my_snapshot
icc_shell> query_qor_snapshot -name my_snapshot -display
```

Figure 4-11 displays a typical view of the Categorized Timing Report Interactive Viewer.

Figure 4-11 Categorized Timing Report Interactive Viewer

The screenshot shows a software interface titled "Highlighted Objects in Layout Window". At the top, there are several filter checkboxes: Path (checked), Buffers (checked), Inverters (checked), Macros (checked), Transition Degradation (unchecked), and Largest Fanout (unchecked). Below this, the "Input file:" is listed as "testcase/snapshot/ss1.brm.max.rpt", "Filters:" as "-wns .0 -zero_path .0 -fanout 40 -transition_degradation 2.0 -slew_degradation 2.0 -logic_levels 50", and "Sort Column:" as "wns (ascending)". The "Number of Paths:" is 40. A table follows, with columns: False Path, Display Path, Path Group, Start Point, End Point, WNS, Zero Path, and Worst Incremental. The table lists five paths: P20, P0, P1, P2, and P3, each with its respective start and end points and timing values.

False Path	Display Path	Path Group	Start Point	End Point	WNS	Zero Path	Worst Incremental
	[X] P20	clock_to	d_0/dar/reg/regx22x	tg_r_data[22]	-109.6170	8.3000	106.7550
	[] P0	clock	d_0/dar/reg/regx22x	d_0/dar/reg/regx23x	-104.4340	12.0470	109.7280
	[] P1	clock	d_0/dar/reg/regx22x	d_0/dar/reg/regx25x	-101.4710	16.3480	106.7550
	[] P2	clock	d_0/dar/reg/regx22x	d_0/sar/reg/regx31x	-99.9470	19.0770	106.7550
	[] P3	clock	d_0/dar/reg/regx22x	d_0/dar/reg/regx31x	-99.7650	19.2450	106.7550

For more information about the Categorized Timing Reporting Interactive Viewer, see IC Compiler Help.

The following sections describe the features commonly used during timing analysis:

- [Applying Filters](#)
- [Creating Subgroups](#)
- [Analyzing the Clock Tree Synthesis Results](#)

Applying Filters

You can apply filters to selected paths in the design by using the following methods:

- Check the filters in the Query QoR Snapshot menu by choosing Timing > Query QoR Snapshot in the GUI.
- Use the `-filters` option of the `query_qor_snapshot` command.

For example, the following command reports all paths that have a worst negative slack less than zero:

```
icc_shell> query_qor_snapshot -name my_snapshot -filters "-wns ,0"
```

The following example reports all paths that meet any of the specified filter requirements:

```
icc_shell> query_qor_snapshot -name my_snapshot \
    -filters "-wns -4.0 -transition_degradation 5.0 \
    -slew_degradation 4.0 -fanout 20"
```

If you do not specify the `-filters` option, the `query_qor_snapshot` command automatically applies a set of default filters. The default filter settings for the maximum condition are shown in the following example:

```
icc_shell> query_qor_snapshot -name my_snapshot \
    -filters "-wns ,0 -zero_path ,0 -fanout 40 \
    -transition_degradation 2.0 -slew_degradation 2.0 -logic_level 50"
```

For a complete list of filters, see the `query_qor_snapshot` man page.

When you enable the top-level view by specifying the `-hierarchy` option, the `query_qor_snapshot` command automatically generates a report of all violating paths between the extracted timing models (ETMs), block abstractions, and hard macros. You can apply filters or expand the top-level paths by specifying the `-to`, `-from`, and `-through` options.

Creating Subgroups

You can create subgroups of selected paths to manage a large number of paths in the categorized timing reports. To create subgroups, specify the `-subgroup` option with the allowed subgroup options of the `query_qor_snapshot` command. You can optionally specify subgroup names. If you do not specify the subgroup names, the command automatically assigns a name to each subgroup, such as Q1 or Q2. A subgroup named Others is created by default, which contains all remaining paths that are not included in the subgroups created by the `-subgroup` option.

The following restrictions apply when you specify the `-subgroup` option:

- The allowed subgroup options to select and sort paths for each subgroup include `-columns`, `-filters`, `-and`, `-group_by`, `-sort_by`, and `-case_sensitive`.
- You cannot specify the `-subgroup` option with the `-hierarchy`, `-incremental`, and `-clock_tree_only` options.

You can specify the `-subgroup` option multiple times for each `query_qor_snapshot` command. Each subgroup consists of paths that meet the requirements specified by the `-filters` option for that subgroup. The paths available to subsequent subgroups are those paths that do not meet the requirements of the options specified by the preceding subgroups.

For example, the following command creates three subgroups, WNS_group, Trans_deg, and an arbitrary subgroup name assigned by the command, in addition to the Others default subgroup.

```
icc_shell> query_qor_snapshot -name my_snapshot \
    -subgroup {"WNS_group" -group_by path_group -sort_by wns \
        -filters "-wns ,0"} \
    -subgroup {"Trans_deg" -sort_by startpoint \
        -filters "transition_deg 2.0"} \
    -subgroup {-filters "-w_deg 2.0"}
```

Analyzing the Clock Tree Synthesis Results

You can also use the Categorized Timing Report Interactive Viewer to analyze the results of clock tree synthesis. For example, the following commands open the viewer with the default filter settings:

```
icc_shell> create_qor_snapshot -name my_snapshot -clock_tree
icc_shell> query_qor_snapshot -name my_snapshot -clock_tree_only -display
```

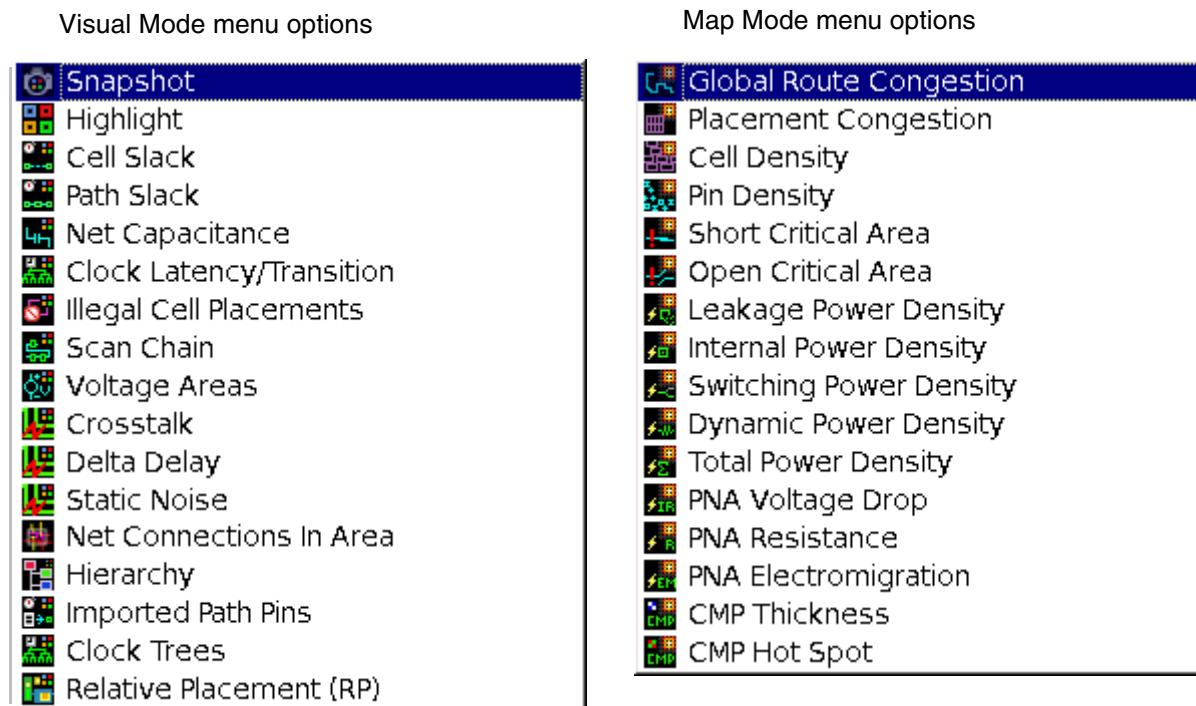
For more information about using the viewer to analyze clock tree synthesis, see [“Generating Categorized Timing Reports for Clock Tree Synthesis” on page 5-140](#).

Analyzing the Placement in the GUI

The IC Compiler GUI provides a wide range of tools for analyzing and visualizing the quality of results after placement. To generate a layout view showing the desired quality-of-results parameter, open the Visual Mode or Map Mode panel (View > Visual Mode or View > Map Mode), and from the pull-down menu in the panel, select the desired type of report.

[Figure 4-12](#) shows the types of visual reports available in the Visual Mode and Map Mode panels.

Figure 4-12 Report Types in the Visual Mode and Map Mode Panels



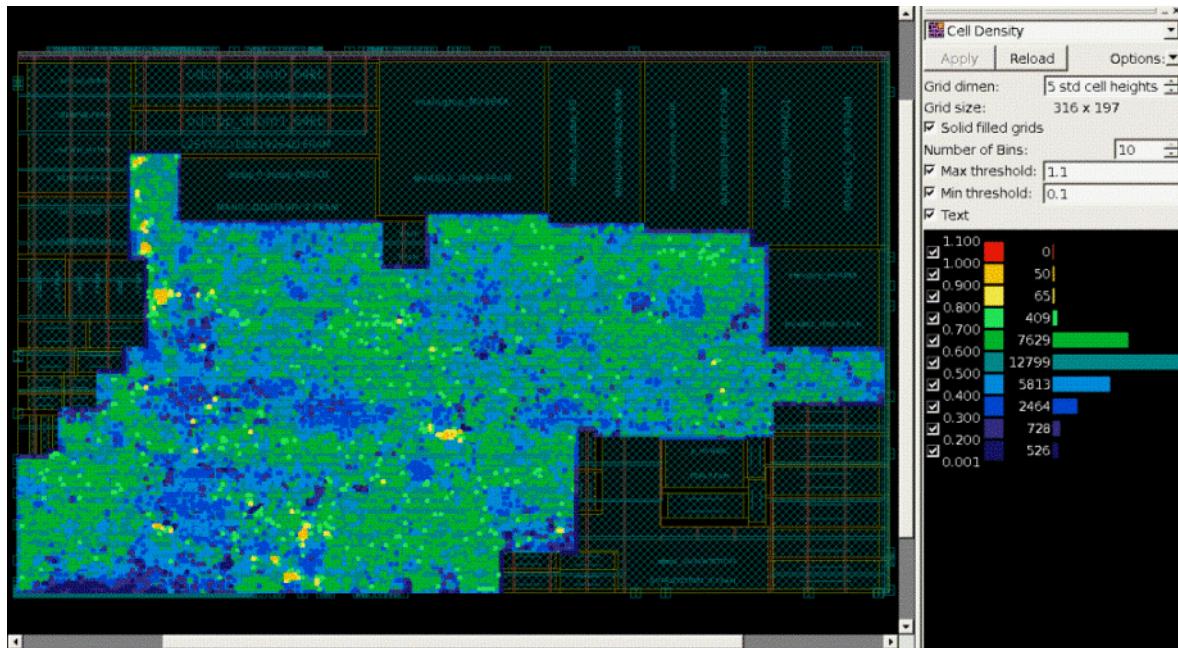
After you select the desired type of visual report, the panel changes to show the options for that type of display. Select the desired options and click the Reload button, which opens a dialog box that typically offers some additional options. In the dialog box, click OK to update the display.

Some of the GUI features commonly used after placement are described in the following sections.

Cell Density Map

A cell density map is a typical GUI report. When you select Cell Density as the Map Mode, the Map Mode panel offers you the option to specify the analysis grid dimensions, solid or unfilled grid units, the number of histogram bins, the bin threshold levels, and the utilization text display. [Figure 4-13](#) shows a typical cell density map.

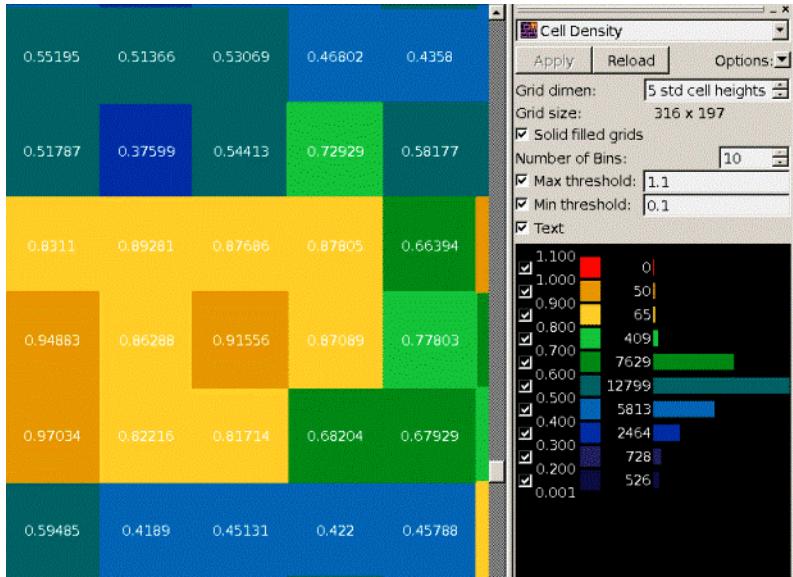
Figure 4-13 Cell Density Map, Full-Chip View



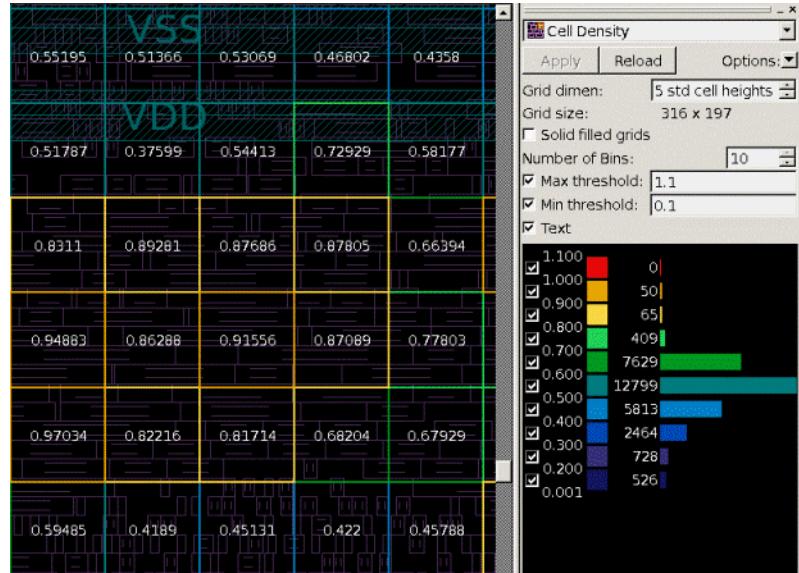
To generate a cell density map, the tool divides the layout into a grid of squares measuring five standard-cell heights on each side. It finds the area utilization in each grid square and then color-codes each square according to the utilization value. It also displays a histogram showing the number of grid squares in each of the utilization value bins. When the view is zoomed in enough, the utilization values are displayed inside the grid squares, as shown in [Figure 4-14](#). Otherwise, the text values are omitted from the display. [Figure 4-14](#) shows both a solid-filled grid display and an unfilled grid display.

Figure 4-14 Cell Density Maps With and Without Solid Filled Grids

Cell Density Map With Solid Filled Grids



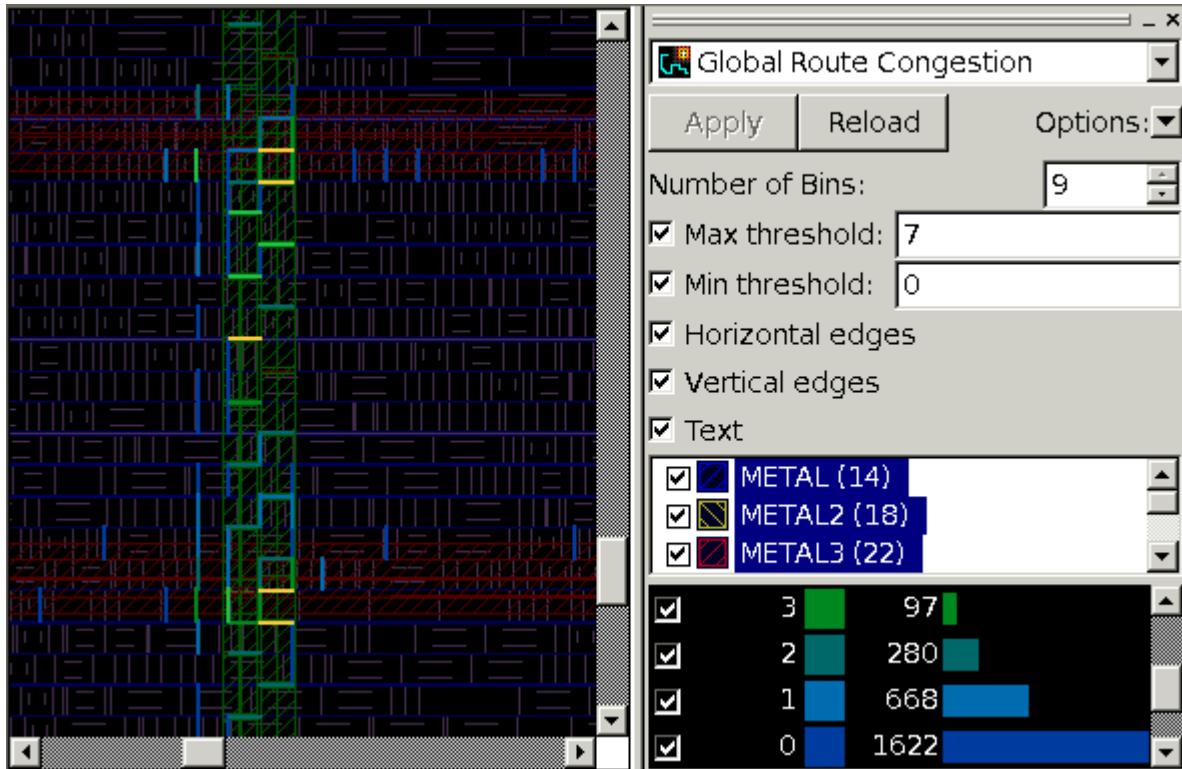
Cell Density Map Without Solid Filled Grids



Congestion Maps

A congestion map can help you visualize the quality of placement with respect to the avoidance of routing congestion. An example is shown in [Figure 4-15](#).

Figure 4-15 Global Route Congestion Map

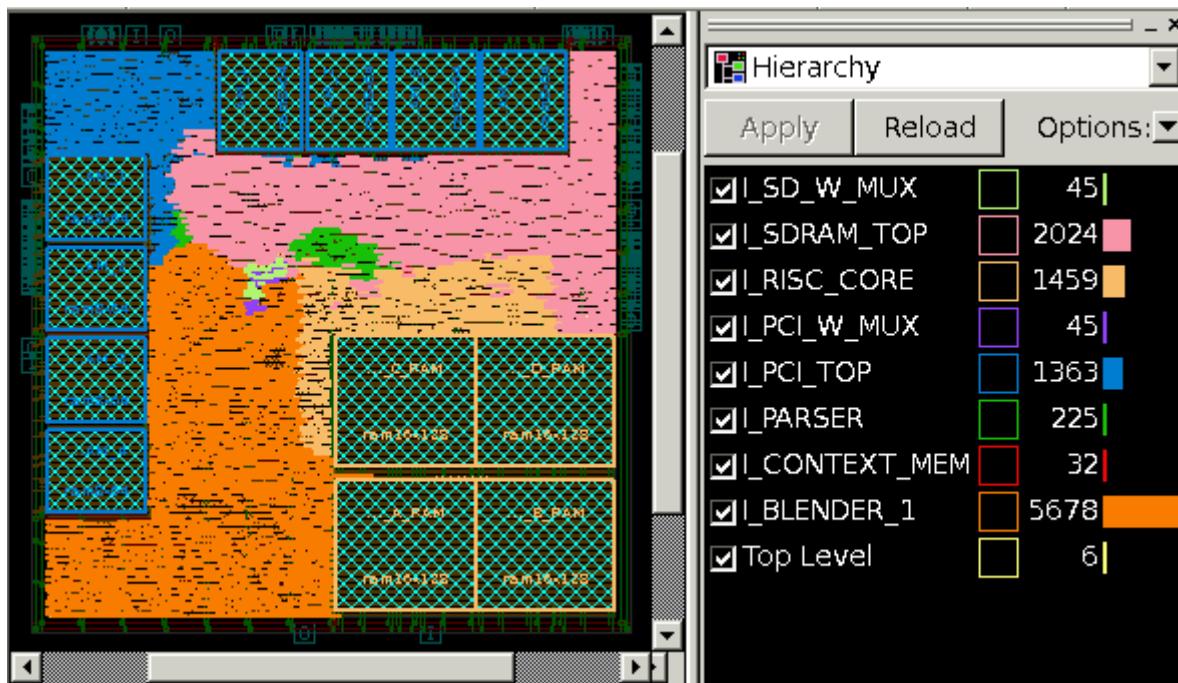


To generate a congestion map, choose **Route > Global Route Congestion Map** in the GUI and click **Reload** in the Map Mode dialog box. You can also generate congestion maps based on the track assignment and detail routing in your design. To generate a congestion map based on track assignment, choose **Route > Track Assign Congestion Map** in the GUI. To generate a congestion map based on detail routing, choose **Route > Detail Route Congestion Map** in the GUI. For more information about the congestion maps, see [“Analyzing Congestion” on page 6-114](#).

Hierarchy Visual Mode

The hierarchy visual mode is a display of the design with cells highlighted and color-coded according to block membership or levels of hierarchy. By default, the layout display shows the cells belonging to each top-level block highlighted in its own color, as shown in [Figure 4-16](#). A histogram-like bar graph shows the number of cells belonging to each top-level block, identified by name.

Figure 4-16 Hierarchy Visual Mode

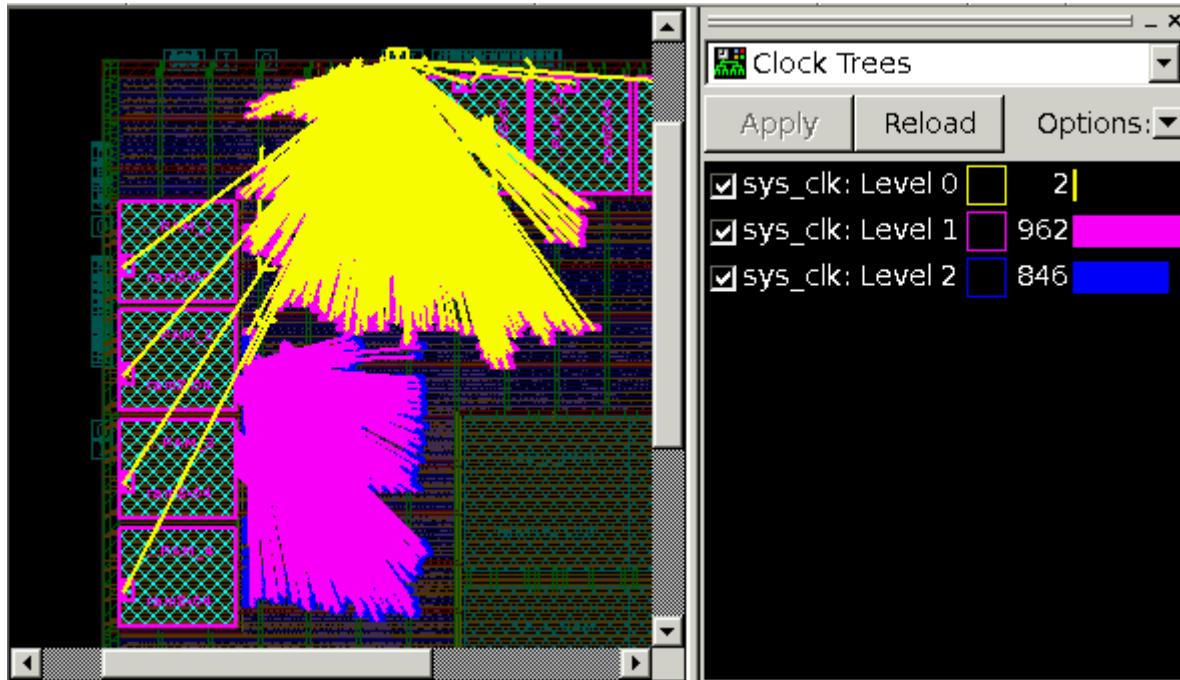


You can also highlight cells using a different color for each second-level hierarchical block or each third-level hierarchical block, and so on, or you can highlight only the cells belonging to one or more particular hierarchical blocks, identified by name, at any level. You can set the hierarchical coloring options in the dialog box opened by clicking the Reload button.

Clock Tree Visual Mode

The clock tree visual mode highlights the flylines and cells of a clock tree using a different color for each buffer level of the tree. For example, [Figure 4-17](#) shows the connections in levels 0, 1, and 2 of the sys_clk clock tree, with the level 0 in yellow, level 1 in magenta, and level 2 in blue. This display can help you visualize the extent and approximate route lengths of the clock tree. A histogram-like bar graph shows the number of objects at each level of the clock tree.

Figure 4-17 Clock Tree Visual Mode

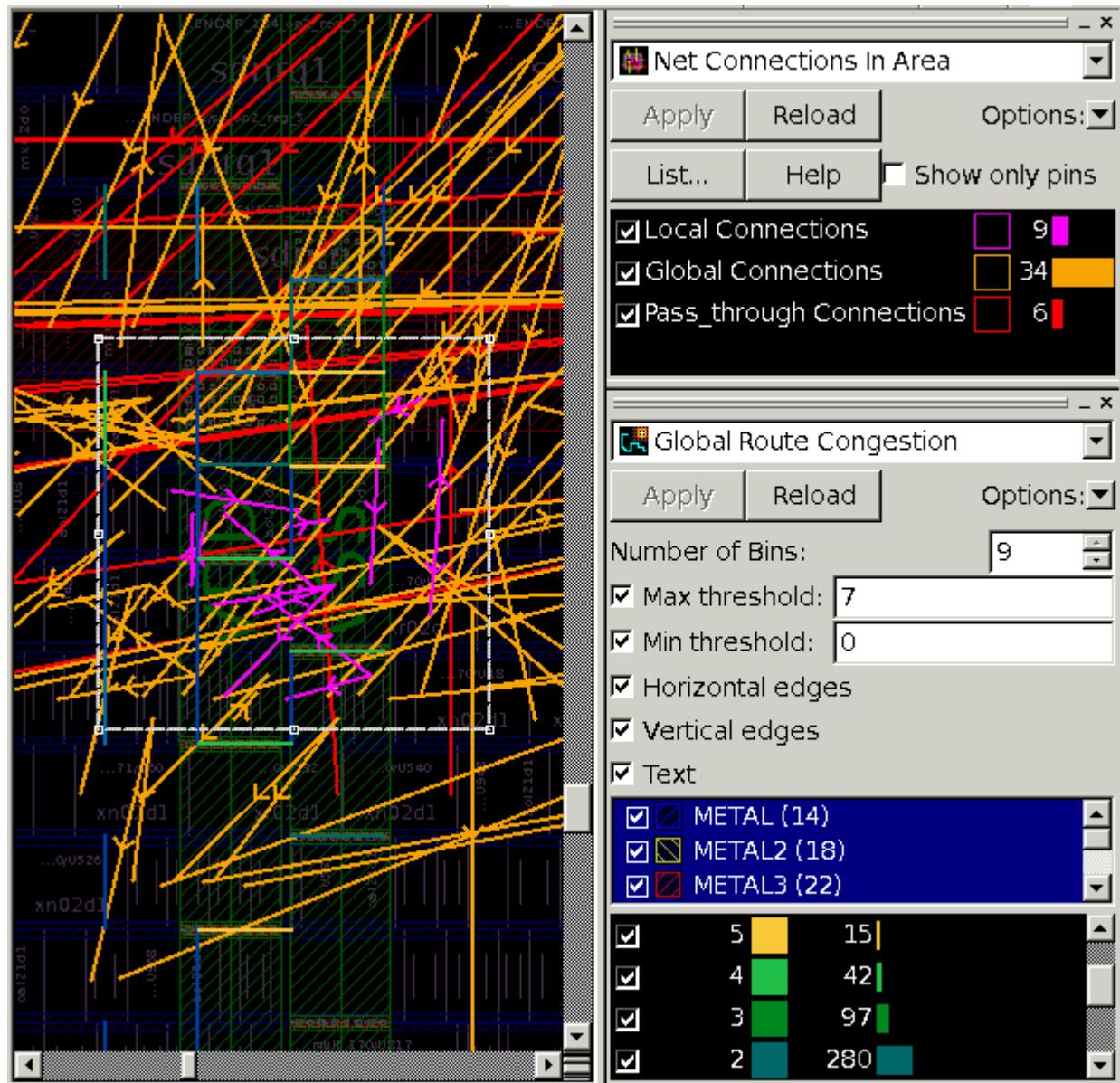


Net Connections in Area Visual Mode

The net connections in area visual mode lets you visualize the nets enclosed in or crossing a specified rectangular area of the design and examine the properties of those nets. To display net connections in an area, first select Net Connections In Area in the Visual Mode panel. Click Reload. In the dialog box, you can enter the coordinates of two corners of the desired area, or you can drag a rectangle directly in the layout view. You can also specify the types of nets to be displayed: signal and clock nets by default, and optionally power, ground, tie-high, and tie-low nets. Then click OK to update the display.

[Figure 4-18](#) shows an example of a net connections in area display superimposed on a global route congestion map. The area of interest is the white dotted rectangle, which was drawn to analyze nets in a region of high congestion. There are three types of flyline connections highlighted in the view, called the local, global, and pass-through connections.

Figure 4-18 Net Connections in Area, Flyline Display

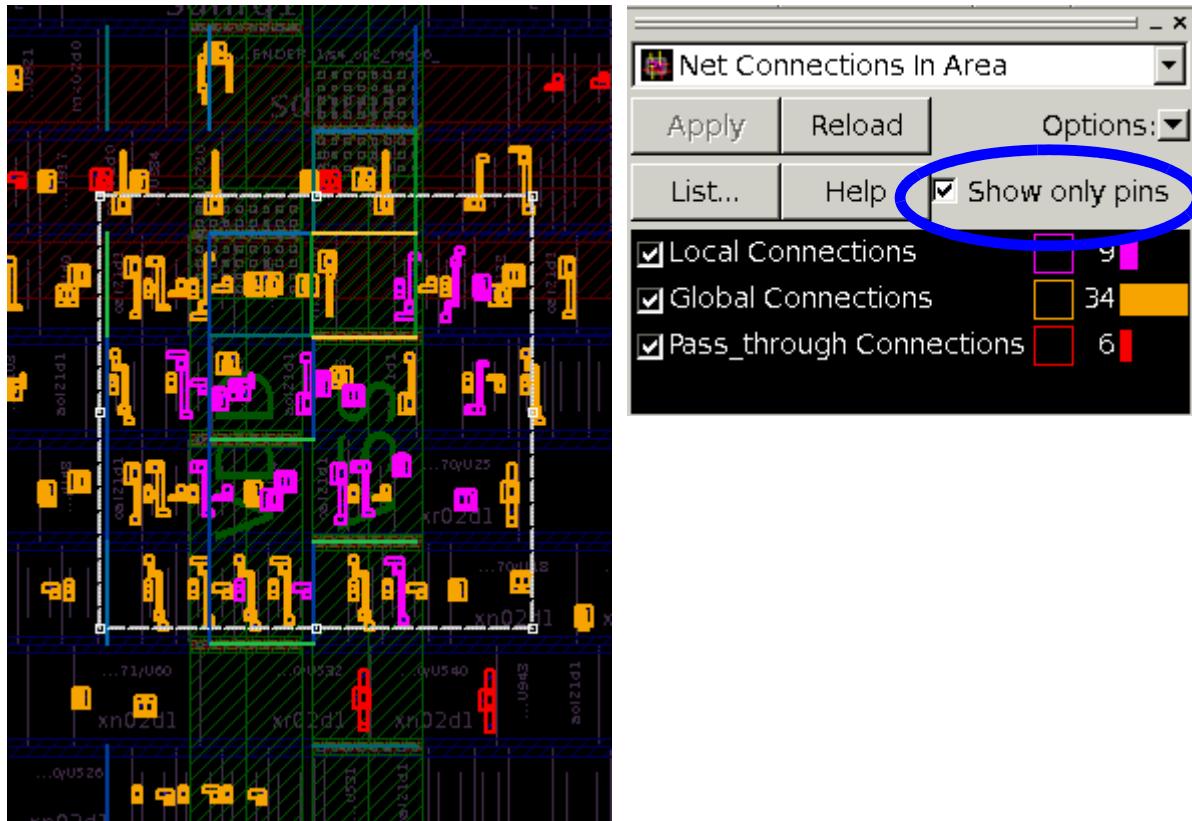


The local connections, shown in magenta, are completely enclosed by the rectangle. The global connections, shown in orange, are the nets that have one or more pins both inside

and outside the rectangle. The pass-through connections, shown in red, have pins outside of the rectangle only; the nets are expected to pass through the rectangle without connecting to any pins inside the rectangle.

You can optionally turn off the display of flylines and show the connection pins instead by checking the “Show only pins” option in the Net Connections In Area panel. See [Figure 4-19](#). You can also separately enable or disable the display of local, global, or pass-through type pins or net connections by checking or unchecking the respective options in the Visual Mode panel.

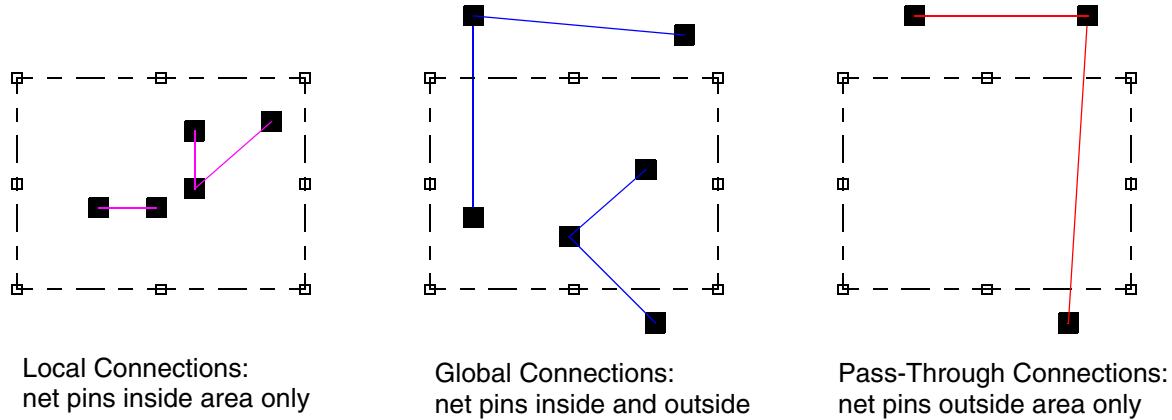
Figure 4-19 Net Connections in Area, Pin-Only Display



The three types of net connections highlighted in the net connections in area view traverse the designated rectangular area in different ways, as demonstrated in [Figure 4-20](#).

Connections are local if all the pins of the net are inside the designated area. Connections are global if one or more pins of the net are both inside and outside the area. Connections are pass-through if all the pins of the net are outside the area and the connections are expected to pass through the area.

Figure 4-20 Local, Global, and Pass-Through Connections



In the net connections in area display, a bar graph shows the numbers of local, global, and pass-through nets. To get more information about these nets in a category, select the category in the bar graph and then click the List button. This opens a table in a new dialog box. The table lists the local and hierarchical net names, net types, number of pins, and other information in the nets. From this table, you can change the types of net information displayed, highlight specific nets in the layout, generate histograms of the net data, and export the data to a spreadsheet.

The net connections in area display capability lets you determine the net topology in a particular region and observe the properties of the nets in that area. This feature works with both routed and unrouted designs. You can analyze the nets that cross a specific area and use the information to change the floorplan or add constraints such as blockages. For example, excessive local nets might imply that utilization changes are needed, or excessive global or pass-through nets might imply that floorplan or placement constraint changes are needed.

Viewing Scan Chains

You can view scan chains either by generating a scan chain report or by viewing the scan chains in the GUI.

To generate a scan chain report, use the `report_scan_chain` command. The scan chain report lists the current scan chain connections. The `report_scan_chain` command displays the chain stub connectivity as it exists in the netlist after a chain has been validated by using the `check_scan_chain` command. All cells in the scan chain are reported, not just scan cells. Note that a SCANDEF file does not necessarily follow the order of connectivity defined in the netlist. Components inside a floating list are considered unordered, and ordered lists are also unordered among the other lists.

To view the scan chains in the GUI,

1. Choose Placement > Color By Scan Chain.

The Visual Mode panel opens.

2. Load the scan chain information by clicking Reload in the Visual Mode panel.

The number it displays next to each scan chain is the total number of cells in each scan chain plus two additional cells for the start and stop points.

3. Select the scan chains that should be highlighted in the layout window.

The layout window displays the scan chains. If you hover the pointer over a scan chain component, a ToolTip pops up and displays the object's properties, including the scan chain to which it belongs.

Fine-Tuning the Placement and Optimization Results

The following topics describe features that you can use to further improve your placement and optimization results:

- [Performing Magnet Placement](#)
- [Using the Top-Level Buffering Technique for Optimization](#)
- [Performing Incremental Physical Optimization](#)
- [Performing Preroute Focal Optimization](#)
- [Refining Placement](#)

Performing Magnet Placement

To improve congestion for a complex floorplan or to improve timing for the design, you can use magnet placement to specify fixed objects as magnets and have the tool move their connected standard cells close to them. You can specify fixed macro cells, a pin of a fixed macro cell, or an I/O pin as the magnet object.

For best results, perform magnet placement before standard cells are placed.

To perform magnet placement, use the `magnet_placement` command with a specification of the magnets and options for any special functions you need to perform.

```
icc_shell> magnet_placement [options] magnet_objects
```

[Table 4-6](#) shows the `magnet_placement` options.

Table 4-6 magnet_placement Command Options

To do this	Use this option
Enable the movement of fixed (or soft fixed) cells and display a warning that lists the fixed cells to be moved.	<code>-move_fixed</code> <code>-move_soft_fixed</code>
Fix (or soft fix) cells in their locations after magnet placement.	<code>-mark_fixed</code> <code>-mark_soft_fixed</code>
Specify the number of logic levels from the magnet that should be checked for magnet placement.	<code>-logical_level number</code>
Prevent movement of buffers and inverters.	<code>-exclude_buffers</code>
Prevent cells from being placed over soft blockages.	<code>-avoid_soft_blockages</code>
Prevent placement beyond sequential cells.	<code>-stop_by_sequential_cells</code>
Specify to pull only the cells between the magnet objects and the specified pin, port, or cell objects on the timing path. This option is mutually exclusive with the <code>-logical_level</code> option.	<code>-stop_points object_list</code>
Specify to pull the specified cells that form a continuous datapath toward the magnet object.	<code>-cells cell_list</code>

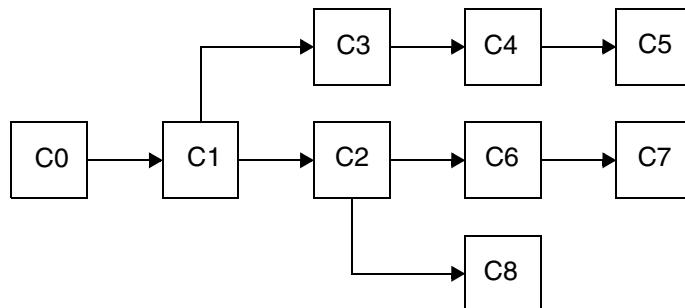
Table 4-6 magnet_placement Command Options (Continued)

To do this	Use this option
Specify not to pull the specified cells toward the magnet object.	<code>-exclude_cells cell_list</code>
Specify that all the cells of a hierarchical module should be pulled toward the magnet object and the mode in which it is performed.	<code>-hierarchy_mode default block all</code>
To instruct the <code>magnet_placement</code> command whether or not to pull cells toward the magnet objects, you can set the <code>magnet_placement_fanout_limit</code> variable to specify the fanout limit. If the fanout of a net exceeds the specified limit, the command does not pull cells of the net toward the magnet objects. The default setting is 1000.	
Magnet placement allows cells to be overlapped by default. To prevent overlapping of cells, you can set the <code>magnet_placement_disable_overlap</code> variable to <code>true</code> , changing it from its default of <code>false</code> .	

Pulling Cells in a Continuous Datapath Toward a Magnet Object

You can pull cells that form a continuous datapath toward a magnet object by specifying the `-cells` option. When you use this option, specify one magnet object only. You cannot use the `-cells` option with the `-stop_points`, `-stop_by_sequential_cells`, `-exclude_buffers`, or `-logical_level` option.

The following examples use the datapaths shown in [Figure 4-21](#).

Figure 4-21 Example of Pulling Cells in a Continuous Datapath

The following command pulls all cells, C1 through C8, toward the C0 magnet object:

```
icc_shell> magnet_placement c0 -cells {C1 C2 C3 C4 C5 C6 C7 C8}
```

The following command pulls the C6, C7, and C8 cells toward the C2 magnet object:

```
icc_shell> magnet_placement c2 -cells {C6 C7 C8}
```

Although the C3, C4, and C5 cells form a continuous datapath, they are isolated from the C0 cell, so the following command pulls no cells toward the C0 cell.

```
icc_shell> magnet_placement c0 -cells {C3 C4 C5}
```

The following command pulls no cells toward the C0 magnet object because the C3, C6, and C8 cells do not form a continuous datapath.

```
icc_shell> magnet_placement c0 -cells {C3 C6 C8}
```

Excluding Cells From Magnet Placement

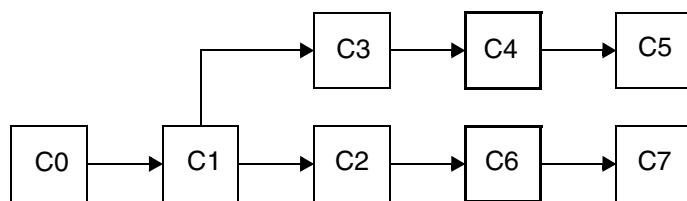
You can specify which cells to skip during magnet placement. To exclude cells from being pulled toward a magnet object, specify the `-exclude_cells cell_list` option with the `magnet_placement` command.

For example, to exclude the C4 and C6 cells from magnet placement in the datapaths shown in [Figure 4-22](#), enter

```
icc_shell> magnet_placement c0 -exclude_cells {C4 C6} -logical_level 4
```

This command pulls the C1, C2, and C3 cells toward the C0 magnet object. The C5 and C7 cells are not pulled because they do not form continuous datapaths when the C4 and C6 cells are excluded.

Figure 4-22 Datapaths Example



To exclude cells when reporting cells that can be pulled toward a magnet object, specify the `-exclude_cells cell_list` option with the `get_magnet_cells` command. Using the same datapaths shown in [Figure 4-22](#), the following command reports the C1, C2, and C3 cells but not the excluded cells, C4 and C6, and the cells in the subsequent logic level, C5 and C7.

```
icc_shell> get_magnet_cells c0 -exclude_cells {C4 C6} -logical_level 4
```

Reporting Magnet Cells

To return a collection of cells that can be moved with magnet placement, use the `get_magnet_cells` command with the options you need.

```
icc_shell> get_magnet_cells [options] magnet_list
```

Table 4-7 get_magnet_cells Command Options

To do this	Use this option
Get the fixed (or soft fixed) cells that can be moved.	<code>-move_fixed</code> <code>-move_soft_fixed</code>
Get the cells in the specified number of logic levels from the magnet.	<code>-logical_level number</code>
Get the cells encountered between magnet cells and sequential cells.	<code>-stop_by_sequential_cells</code>
Exclude the buffers and inverters.	<code>-exclude_buffers</code>
Identify the cells between magnet objects and the specified pin, port, or cell objects on the timing path. This option is mutually exclusive with the <code>-logical_level</code> option.	<code>-stop_points object_list</code>
Exclude the specified cells when reporting cells that can be pulled toward the magnet object.	<code>-exclude_cells cell_list</code>

Using the Top-Level Buffering Technique for Optimization

If your design contains the following characteristics, the traditional place and route flow might not produce optimal results:

- A fragmented floorplan with thin channels caused by many macros and blocks
- Minimal or no interface logic between blocks
- Buffering that is driven by design rule constraints, primarily the maximum net length constraint
- Many long nets through channels, especially low fanout or point-to-point nets

To improve the QoR and reduce congestion of such designs, you should consider using the top-level buffering technique for optimization, as shown in the following example:

```
icc_shell> set_max_net_length integer
icc_shell> create_placement
icc_shell> create_buffer_tree -align_hierarchy_for_long_nets
....
icc_shell> create_buffer_tree -on_route
```

When you use this technique,

- You must set the `max_net_length` constraint before running the `create_buffer_tree` command.
- Run the `create_placement` command only if your design contains interface logic at the top level.
- Run the `create_buffer_tree -align_hierarchy_for_long_nets` command to perform port punching and buffering of the nets which exceed the `max_net_length` constraint.
- If the design requires further optimization at the top-level, run the `place_opt -skip_initial_placement` or `psynopt` command after running the initial `create_buffer_tree` command.
- If further optimization is not required between the `create_buffer_tree -align_hierarchy_for_long_nets` and `create_buffer_tree -on_route` commands, combine them into a single step by running the `create_buffer_tree -align_hierarchy_for_long_nets -on_route` command.
- The `create_buffer_tree -on_route` command performs topology-based DRC fixing and routing. However, if you want to perform further optimization before routing, run the `create_buffer_tree -on_route -skip_detail_route` command instead.

Performing Incremental Physical Optimization

You can run incremental placement-based optimization that supports area recovery, design rule fixing, sizing, and route-based optimization by using the `psynopt` command. By default, this command performs timing optimization and design rule fixing, based on the maximum capacitance and maximum transition settings while keeping the clock networks untouched. It can also perform power optimizations. The `psynopt` command continues to optimize until no more optimizations can be performed. When done, it completes the optimizations with a legalized placement of the design.

This process can remove dangling cells (cells that do not drive other cells), resulting in a reduced number of cells in the design. If you need to prevent the removal of such cells, do one of the following:

- To prevent the removal of specific cells, use the `set_dont_touch` command to apply a `dont_touch` attribute on each of those cells.
- To retain the unloaded cells globally, set the `physopt_delete_unloaded_cells` variable to `false`.

[Table 4-8](#) shows the options supported by the `psynopt` command.

Table 4-8 The psynopt Options

To do this	Use this option
Enable area recovery within the cluster boundary for noncritical paths. Using this switch can have a severe runtime impact if the design contains many high-fanout nets.	<code>-area_recovery</code>
Perform only area recovery. You cannot use this option with the <code>-only_design_rule</code> or <code>-only_hold_time</code> options.	<code>-only_area_recovery</code>
Exit <code>psynopt</code> without fixing design rule violations, allowing you to check the results in a constraint report before fixing the violations.	<code>-no_design_rule</code>
You cannot use the <code>-no_design_rule</code> option with the <code>-only_design_rule</code> option.	
Perform design rule fixing without performing timing optimizations. This option uses <code>psynopt</code> placement to target DRC problems.	<code>-only_design_rule</code>
You cannot use the <code>-only_design_rule</code> option with the <code>-no_design_rule</code> or <code>-only_area_recovery</code> options.	
Fix only hold time violations; ignore other design rules. The <code>set_fix_hold</code> command must be specified for hold time fixing to be performed.	<code>-only_hold_time</code>
You cannot use this option with the <code>-no_design_rule</code> or <code>-only_area_recovery</code> options.	
Restrict postroute flow optimization to gate sizing only. You cannot use this option with the <code>-in_place_size_only</code> option.	<code>-size_only</code>
Restrict postroute flow optimization to in-place gate sizing only. You cannot use this option with the <code>-size_only</code> option.	<code>-in_place_size_only</code>
Restrict cell sizing to cells having the same footprint and size.	<code>-preserve_footprint</code>

Table 4-8 The psynopt Options (Continued)

To do this	Use this option
Refine critical paths by performing register optimization.	-refine_critical_paths
Reduce congestion to improve routability.	-congestion
Enable power optimization.	-power
Enable only power optimization.	-only_power
Ignore the scan chain connections during the congestion optimization stage. By default, the command considers scan nets during congestion optimization (<code>psynopt -congestion</code>).	-ignore_scan
Continue placement when the design contains scan chains but no SCANDEF data. By default, missing SCANDEF data causes the command to exit with an error message. Setting this option enables the placer to continue with a warning and results in reduced QoR.	-continue_on_missing_scandef

Performing Preroute Focal Optimization

You can use the `preroute_focal_opt` command to perform targeted preroute optimization on a placed design to fix setup, hold, or logical DRC violations or to perform incremental high-fanout synthesis, layer optimization, or cell sizing.

For each run, you must specify the focus of the optimization (setup, hold, logical design rule constraints, high-fanout nets, layer optimization, or sizing):

- To fix setup violations, use the `-setup_endpoints` option.

To fix all setup violations, set this option to `all`. To fix specific setup violations, specify the endpoints by using an endpoint file or by providing a collection that contains the endpoints. For information about the format of the endpoint file, see “[Endpoint File Format](#)” on page 4-88.

- To fix hold violations, use the `-hold_endpoints` option.

To fix all hold violations, set this option to `all`. To fix specific hold violations, specify the endpoints by using an endpoint file or by providing a collection that contains the endpoints. For information about the format of the endpoint file, see “[Endpoint File Format](#)” on page 4-88.

- To fix logical DRC violations on specific pins, use the `-drc_pins` option.
To fix DRC violations on all pins, set this option to `all`. To fix DRC violations on specific pins, specify the pins by using an endpoint file or by providing a collection that contains the pins. For information about the format of the endpoint file, see “[Endpoint File Format](#)” on page 4-88.
- To fix logical DRC violations on specific nets, use the `-drc_nets net_file` option.
For information about the format of the net file, see “[Net File Format](#)” on page 4-88.
- To perform incremental high-fanout synthesis, use the `-high_fanout_nets` option.
When you specify this option, the `preroute_focal_opt` command performs incremental high-fanout synthesis on all nets that meet the high-fanout threshold. It also fixes the logical DRC violations on the nets created after high-fanout synthesis.
- To perform layer optimization, use the `-layer_optimization` option.
In advanced technologies, the resistance values on upper layers are much smaller than on lower layers. If the design contains an existing buffer tree, you can use this option to improve DRC violations and timing by using the upper layers for routing the buffer tree. For details about using the `preroute_focal_opt` command for layer optimization, see “[Performing Layer Optimization on Existing Buffer Trees](#)” on page 4-89.
- To restrict the optimization to cell sizing, and to select the mode in which to perform sizing, use the `-size_only_mode` option with the appropriate setting:
 - To perform cell-density-based sizing, set the `-size_only_mode` option to `density`. The default density setting is 0.95. You can change the density threshold by using the `psynopt_density_limit` variable.
 - To perform in-place sizing, set the `-size_only_mode` option to `size_only`.
 - To preserve the cell footprint during sizing, set the `-size_only_mode` option to `footprint`.

Note:

Although the `preroute_focal_opt` command targets a single metric in each run, legalization and extraction effects might cause timing or DRC changes on untargeted objects.

By default, the `preroute_focal_opt` command uses medium effort to fix the violations. When using medium effort, the `preroute_focal_opt` command explores more solution spaces while fixing the violations, including solutions that exceed the density limit. If medium effort does not fix the violations, you can use high effort (the `-effort high` option), which explores even more solution spaces and uses runtime-expensive accurate signal integrity reestimation. Due to the additional runtime required for high effort, you should use this only when you have a few remaining violations to close.

If you find that the `preroute_focal_opt` command leaves unfixed violations, you can enable verbose reporting during optimization by setting the `preroute_opt_verbose` variable. This variable uses bitwise operation to enable various reporting capabilities. For details about setting this variable, see “[Outputting Additional Information During Preroute Optimization](#)” on page 4-44.

Endpoint File Format

To fix setup, hold, or logical DRC violations on specific endpoints, you can specify the endpoints in an endpoint file. This file can be either uncompressed or compressed in gzip format. When the file is a gzip file, it must have the .gz file extension.

The following formats are supported for specifying an endpoint:

- Endpoint only

```
I_STACK_TOP/I3_STACK_MEM/Stack_Mem_reg_2__1_/D
```

- Endpoint with a slack value

```
I_STACK_TOP/I3_STACK_MEM/Stack_Mem_reg_2__1_/D 0.58      0.44 r      -0.14
```

When you use this format, the tool ignores the slack information and uses only the pin information.

The best way to generate these lines is by editing the file generated by the `report_constraint` command.

Net File Format

To fix logical DRC violations on specific nets, you must specify the nets in a net file.

The following formats are supported for specifying a net:

- Net name only

```
I_STACK_TOP/n342
```

- Net name with a violation

```
I_STACK_TOP/n342 0.70      0.89      -0.19 (VIOLATED)
```

When you use this format, the tool ignores the slack information and uses only the net information.

The best way to generate these lines is by editing the file generated by the `report_constraint` command.

Performing Layer Optimization on Existing Buffer Trees

In advanced technologies, the resistance values on upper layers are much smaller than on lower layers. Taking advantage of the differences in resistance values to reduce buffer counts and possibly improve QoR is referred to as layer optimization. To perform layer optimization on an existing buffer tree, run the `preroute_focal_opt` command with the `-layer_optimization` option.

Because the layer optimization uses the layer assignments and routing rules during RC estimation, you must set any nondefault routing rules before running the `preroute_focal_opt -layer_optimization` command.

By default, the `preroute_focal_opt` command uses the following settings when performing layer optimization:

- Uses the first (topmost) available routing layer as the maximum routing layer
- Uses the second available routing layer as the minimum routing layer
- Processes all buffer trees with more than four buffers (the buffer threshold)
- Uses low congestion effort

To change these default settings, use the `set_preroute_focal_opt_strategy` command.

- To change the maximum routing layer, use the `-max_layer` option.
- To change the minimum routing layer, use the `-min_layer` option.
- To change the buffer threshold, use the `-buffer_threshold` option.
- To change the congestion effort level, use the `-congestion_effort` option. Higher effort levels result in fewer nets being assigned to the top layers, so fewer buffers are removed.
- To change all settings back to their defaults, use the `-default` option.

To report the layer optimization settings used by the `preroute_focal_opt` command, use the `report_preroute_focal_opt_strategy` command.

Refining Placement

If your design shows large timing or violations after you run the `place_opt` command, adjust the `place_opt` options and rerun `place_opt`, as described in “[Performing Placement and Optimization](#)” on page 4-29.

If your design shows small timing or violations after you run `place_opt`, run `psynopt` to fix these violations, as described in “[Performing Incremental Physical Optimization](#)” on page 4-84.

If your design has congestion violations after you run `place_opt`, rerun `place_opt` with high-effort congestion reduction (`-congestion` option). If your design still has congestion violations, you can refine the placement to fix these violations.

To refine the placement, use the `refine_placement` command (or choose Placement > Refine Placement in the GUI). The `refine_placement` command performs incremental placement and legalization.

[Table 4-9](#) describes the available `refine_placement` options.

Table 4-9 refine_placement Options

GUI object	Command option	Description
“Congestion optimization effort” option	<code>-congestion_effort low medium high</code>	Improves the quality of results or reduce runtime. Higher effort levels use additional runtime in an attempt to improve the quality of results. The default is medium.
“Perturbation level effort” option	<code>-perturbation_level min medium high max</code>	
“Whole chip” option	N/A	Performs incremental placement and legalization on the entire chip.
“Specified region” option plus the “Bound coordinates” list	<code>-coordinates {{llx1 lly1 urx1 ury1} {llx2 lly2 urx2 ury2}}</code>	Performs incremental placement and legalization on the specified regions. You can specify either a single region or multiple regions.

Table 4-9 refine_placement Options (Continued)

GUI object	Command option	Description
“Ignore scan-chain connections during placement” check box	<code>-ignore_scan</code>	Ignores the scan chain connections during placement. By default, the command considers scan nets.
“Continue with missing scan definitions” check box	<code>-continue_on_missing_scandef</code>	Specifies to continue placement when the design contains scan chains but no SCANDEF data. By default, missing SCANDEF data causes the command to exit with an error message. Setting this option enables the placer to continue with a warning and results in reduced QoR.

To refine the placement with a minimum impact on QoR, use the medium-effort congestion removal:

```
icc_shell> refine_placement -congestion_effort medium
```

To achieve the best congestion removal, use high-level perturbation, and then perform physical optimization:

```
icc_shell> refine_placement -perturbation_level high
icc_shell> psynopt
```


5

Clock Tree Synthesis

This chapter provides detailed information about the IC Compiler clock tree synthesis capability.

This chapter contains the following sections:

- Prerequisites for Clock Tree Synthesis
- Handling Multivoltage Designs
- Handling Hierarchical Designs
- Analyzing the Clock Trees
- Defining the Clock Trees
- Specifying Clock Tree Exceptions
- Handling Integrated Clock-Gating Cells
- Balancing Multiple Clocks
- Specifying the Clock Tree References
- Specifying Clock Tree Synthesis Options
- Specifying Clock Tree Optimization Options
- Inserting User-Specified Clock Trees
- Handling Existing Clock Trees

- [Verifying the Clock Trees](#)
- [Implementing Clock Trees](#)
- [Implementing Clock Meshes](#)
- [Implementing Multisource Clock Trees](#)
- [Analyzing the Clock Tree Results](#)
- [Fine-Tuning the Clock Tree Synthesis Results](#)
- [Analyzing and Refining the Design](#)

Prerequisites for Clock Tree Synthesis

Before you run clock tree synthesis, ensure that your design and libraries meet the prerequisites described in the following sections.

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

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 5-43.

- 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.

Handling Multivoltage Designs

Before you run clock tree synthesis on multivoltage designs,

- Run the `check_mv_design` command and fix any violations. There should be no isolation and level-shifter violations on the clock nets.
- If the design requires always-on cells, include them when you specify clock tree references by using the `set_clock_tree_references` command. The tool uses the always-on clock tree references only when required.

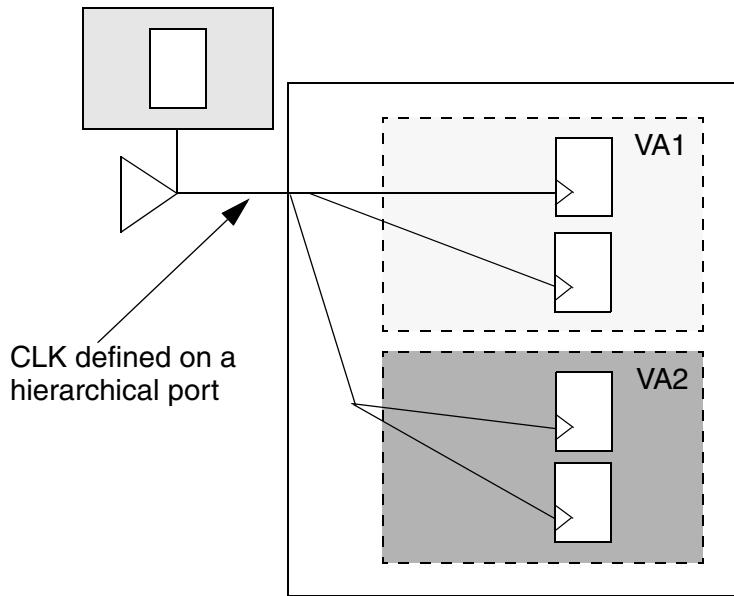
When you run clock tree synthesis on multivoltage designs,

- Sink pins are separated and clustered by voltage area so that clock subtrees are built for each voltage area.
- Guide buffers are inserted on nets driving loads in multiple voltage area to ensure that any subsequent levels of clustering do not mix pins from different voltage areas.
- Supply-net-aware always-on synthesis is performed to buffer the clock nets using dual-rail or single-rail cells, as appropriate. If a net has `dont_touch` attribute due to multivoltage reasons, it is honored.

When buffering clock nets, the tool creates dummy hierarchies, if necessary. Creating dummy hierarchies reduces the need for adding new ports to existing hierarchies. To disable the creation of dummy hierarchies, set the `cts_mv_dummy_hierarchy` variable to `false`.

- Clocks defined on hierarchical pins that drive clock sinks in multiple voltage areas are supported, as shown in [Figure 5-1](#).

Figure 5-1 Hierarchical Clock Pin Driving Clock Sinks in Multiple Voltage Areas



After the clock subtrees are built for each voltage area, clock tree synthesis can proceed in the usual manner, joining the subtrees at the root of the clock net. In addition to the synthesis of the initial clock tree, the preceding behaviors are honored by all clock tree optimization techniques, such as buffer relocation, buffer sizing, gate relocation, gate sizing, and delay insertion.

For more information about working with multivoltage designs, see [Chapter 12, “Multivoltage Design Flow.”](#)

Handling Hierarchical Designs

You can use hierarchical models, such as block abstractions, to increase the capacity and reduce the runtime for top-level clock tree synthesis. Before creating hierarchical models for use with top-level clock tree synthesis, you must perform clock tree synthesis on the blocks.

During clock tree synthesis and optimization, the IC Compiler tool

- Identifies any hierarchical models inside a clock tree
- Honors clocks or generated clocks defined on a port of a hierarchical model or a pin internal to the hierarchical model
- Times the clock subtrees inside the hierarchical model to calculate the phase and transition delays for the hierarchical model

The tool preserves all nets inside the hierarchical model, and the phase delays of the pins of the hierarchical model are calculated for delay balancing.

If there are multiple subtrees after a hierarchical model, the tool synthesizes each subtree independently and does not balance the insertion delay between them, which can result in large skew between them. To reduce this skew, run the `optimize_clock_tree` command after performing clock tree synthesis.

- Honors explicit stop pins, exclude pins, and sink pins on a port of a hierarchical model or inside a hierarchical model

For more information about hierarchical models, see [Chapter 10, “Using Hierarchical Models.”](#)

Handling Extracted Timing Models

An extracted timing model (ETM), which is generated from a gate-level netlist of a design in PrimeTime, can be used to replace the netlist for timing analysis at a higher level of the design. This type of model consists of clock-related arcs and timing arcs constraining input pins and output pins.

Clock tree synthesis and optimization provides the following capabilities to support ETMs:

- Minimize skew across multiple outputs of an ETM for the clock that launches these outputs, taking into account the arc delays in the ETM.
- Honor the clock tree synthesis options and references defined on the internal pins of ETMs.
- Run all clock tree synthesis commands for clocks that start from or go through ETMs.

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

The IC Compiler tool 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 5-79](#).

- 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 the tool 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 5-135](#).

Defining the Clock Trees

The tool 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:

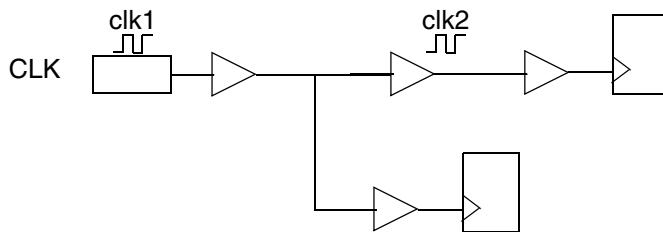
The tool does not synthesize a clock tree if its source is set to a constant value by using `set_case_analysis`.

In addition to simple clock trees, the tool also supports cascaded clock trees. A cascaded clock tree 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 5-2](#), the tool 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 5-2 Cascaded Clock With Two Source Clocks

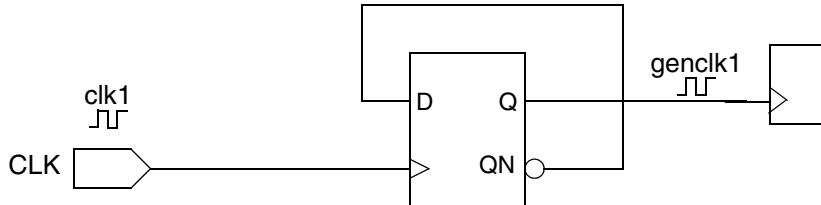


To verify that the clock sources are correctly defined, use the `check_clock_tree` command.

Cascaded Generated Clocks

If a nested clock tree has a generated source, as shown in [Figure 5-3](#), the tool 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 5-3 Cascaded Clock With Generated Clock



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

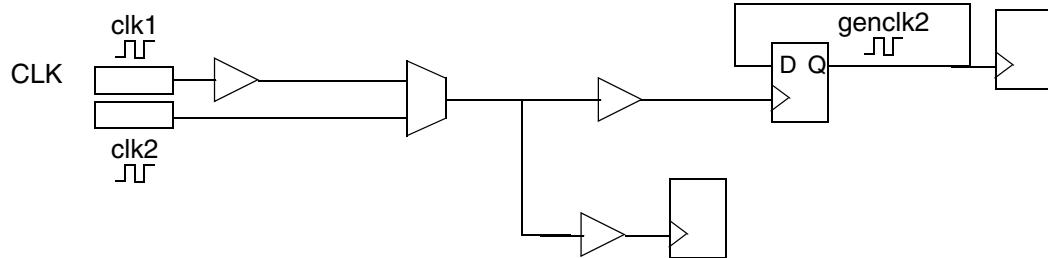
- If the tool cannot trace back to the master-clock source, the tool cannot balance the sinks of the generated clock with the sinks of its source.
- If the master-clock source is not a clock source defined by the `create_clock` or `create_generated_clock` command, the tool cannot synthesize a clock tree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

Use the `check_clock_tree` command to verify that your master-clock sources are correctly defined.

If a nested clock tree has a generated source that is defined at the intersection of overlapping clocks, the tool traces back to the master clock source from which the generated clock is derived and considers the source pin of the generated clock on a per clock basis during synthesis. By default, IC Compiler clock tree synthesis supports querying phase delay and clock tree exceptions for each clock domain to achieve the best QoR for cascaded generated clocks with overlapping clocks.

For example, assume that a flip-flop inside the clk1 and clk2 overlapping domains has a generated clock, genclk2, which is defined at its Q output and which uses clk2 as the master clock. If you perform clock tree synthesis on clk2, the clock tree after that flip-flop is synthesized and balanced as a generated clock of clk2, as is shown in [Figure 5-4](#).

Figure 5-4 Cascaded Generated Clock With Overlapping Clock Domains

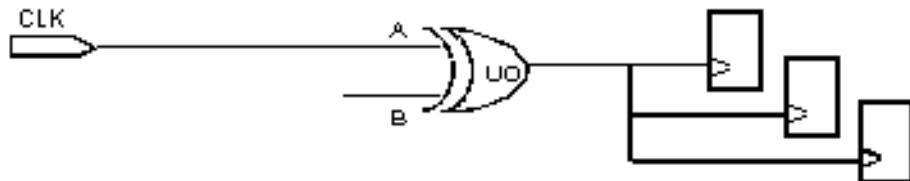


Non-Unate Gated Clocks

If the clock-gating logic uses a non-unate cell, such as an XOR or XNOR gate, the tool 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 5-5](#).

Figure 5-5 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
```

Identifying the Clock Tree Endpoints

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, the tool 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. The tool uses exclude pins only in calculations and optimizations for design rule constraints.

The IC Compiler tool traces the transitive fanout from the clock roots to determine the implicit stop pins (the default sink pins) and the implicit exclude pins.

When the tool finds a pin in the clock's transitive fanout that is defined as a clock pin of either a sequential cell (a latch or flip-flop) or a macro cell, it adds that pin to the default set of clock sinks (unless the fanout of that cell drives a generated clock).

If the fanout of a sequential cell drives a generated clock, the tool considers the clock pin to be an implicit nonstop pin and traces through the sequential cell to locate the clock tree endpoints. In addition, the tool considers the clock input pins of integrated clock-gating (ICG) cells to be implicit nonstop pins.

The IC Compiler tool defines the following clock endpoints as implicit exclude pins:

- Source pins of clock trees in the fanout of another clock
- Nonclock input pins of sequential cells
- Multiplexer select pins
- Three-state enable pins
- Output ports
- Incorrectly defined clock pins (for example, the clock pin does not have trigger edge information or does not have a timing arc to the output pin)
- Buffer or inverter input pins that are held constant (by using `set_case_analysis`)
- Input pins of combinational cells or integrated clock-gating cells that do not have any fanout or that do not have any enabled timing arcs

Verify that the default sink pins (implicit stop pins), implicit nonstop pins, and implicit exclude pins are accurate by generating a clock tree exceptions report, as described in “[Generating Clock Tree Reports](#)” on page 5-135.

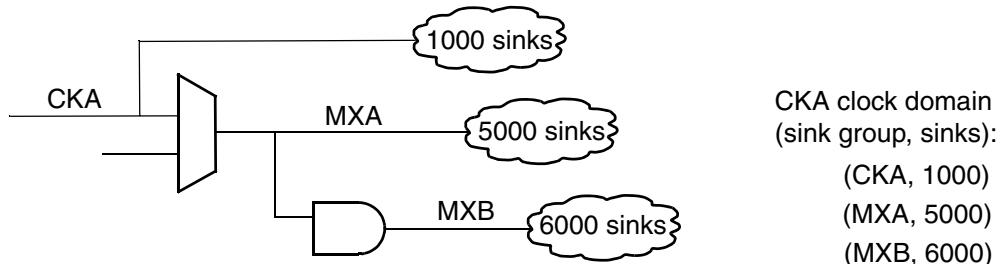
If the default sink pins, implicit nonstop pins, and implicit exclude pins are correct, you are done with the clock tree exception definition. Otherwise, first identify any timing settings, such as disabled timing arcs and case analysis settings, that affect the clock tree traversal. To identify disabled timing arcs in your design, use the `report_disable_timing` command. To identify case analysis settings in your design, use the `report_case_analysis` command. Remove any timing settings that cause an incorrect clock tree definition.

You can modify the set of sink (stop) pins, nonstop pins, and exclude pins by setting clock tree exceptions, as described in “[Specifying Clock Tree Exceptions](#)” on page 5-17.

Analyzing Clock Sink Groups

A clock sink group is a group of clock sinks driven directly by a single net. The sink group assumes the net name. As shown in [Figure 5-6](#), the CKA net drives 1000 sinks, the MXA net drives 5000 sinks, and the MXB net drives 6000 sinks to form three clock sink groups, CKA, MXA, and MXB.

Figure 5-6 Example of Clock Sink Groups



Sink groups can have timing relationships when an endpoint in a sink group has one or more startpoints or endpoints in another sink group. Each startpoint-and-endpoint pair forms one timing relationship path. To better understand the clock structure, you can report sink groups and their timing relationships to guide you in constraining the timing paths between sink groups before running clock tree synthesis. Analyzing sink groups at an early design stage can help balance the timing between sink groups during clock tree synthesis.

To report sink groups and their timing relationships, specify the `-sink_group` option with the `report_clock_tree` command.

Sink groups can overlap because each sink group can be in more than one clock domain. When sink group overlapping occurs, the `report_clock_tree -sink_group` command issues a warning message by default. To suppress the warning message, set the

`timing_enable_multiple_clocks_per_reg` variable to `true`, changing it from its default of `false`.

The following sections describe how to control the sink group reporting options:

- [Setting Sink Group Options](#)
- [Ignoring Clock Tree Exceptions](#)
- [Ignoring Buffers and Inverters](#)

Setting Sink Group Options

When you specify the `-sink_group` option of the `report_clock_tree` command or choose Report Clock Tree > Sink Group in the GUI and select the Sink Group tab, you can also set the options described in [Table 5-1](#) to refine the reporting of sink groups.

Table 5-1 Options for Refining Sink Group Reports

Command option (GUI option)	Description
<code>-clock_trees clock_tree_list</code> ("Clock tree names" text box)	Reports the specified clock trees only. By default, the command reports all currently defined clock trees.
<code>-sink_group_ignore_cts_exceptions</code> ("Ignore cts exceptions" check box)	Ignores the clock tree exceptions, but excludes the nonstop pin exception, when traversing the clock trees to find the sink groups. For more information, see "Ignoring Clock Tree Exceptions" on page 5-14 .
<code>-sink_group_timing_relationship_limit limit</code> ("Timing Relationship limit" text box)	Reports sink groups that have timing relationships with other sink groups that are less than or equal to the specified limit. If you set the limit to zero, the command reports only sink groups without any timing relationships. By default, the limit is infinite, and the command reports all sink groups with timing relationships.
<code>-sink_group_sort_by clock sink_group</code> ("Clock" or "Sink group" option)	Sorts the first-level text output either by clock or by sink group. When you specify <code>clock</code> , which is the default, the sort key is the sink number under the clock. When you specify <code>sink_group</code> , the sort key is the timing relationship number. A tie is broken by the clock or sink group name.

Table 5-1 Options for Refining Sink Group Reports (Continued)

Command option (GUI option)	Description
<code>-sink_group_sort_order ascending descending</code> ("Ascending" or "Descending" option)	Reports sink groups in ascending or descending order. The default is ascending.
<code>-sink_group_detail_file file_name</code> ("Detail output file" text box)	Writes the details of sink group information to the specified file.
<code>-sink_group_timing_relationship_file file_name</code> ("Timing relationship output file" text box)	Writes the timing relationships of the sink pairs in the design, but not the sink group information. Depending on the design, the file size could be large, so use this option with caution.
<code>-sink_group_ignore_buf_inv</code> ("Ignore buffers/inverters" check box)	Treats buffers and inverters as transparent cells during analysis if no clocks or generated clocks are defined on them. For more information, see " Ignoring Buffers and Inverters " on page 5-15.
<code>-sink_group_ignore_icg</code> ("Group ignore ICG" check box)	Treats integrated clock-gating cells as transparent cells during analysis if no clocks or generated clocks are defined on them.

Ignoring Clock Tree Exceptions

By default, the `report_clock_tree -sink_group` command respects the clock tree exceptions when traversing the clock tree network. To ignore the clock tree exceptions, specify the `-sink_group_ignore_cts_exceptions` option. [Table 5-2](#) shows how the command validates a sink with and without this option.

Table 5-2 Validating Sinks With and Without Clock Tree Exceptions

Clock tree exception	Default	Ignoring clock tree exceptions (<code>-sink_group_ignore_cts_exceptions</code>)
Default sink	Counted	Counted
Explicit stop pin	Counted	Ignored
Float pin	Counted	Ignored
Explicit ignore pin	Not counted	Ignored

Table 5-2 Validating Sinks With and Without Clock Tree Exceptions (Continued)

Clock tree exception	Default	Ignoring clock tree exceptions (-sink_group_ignore_cts_exceptions)
Implicit ignore pin	Not counted	Ignored
Nonstop pin	Ignored	Ignored

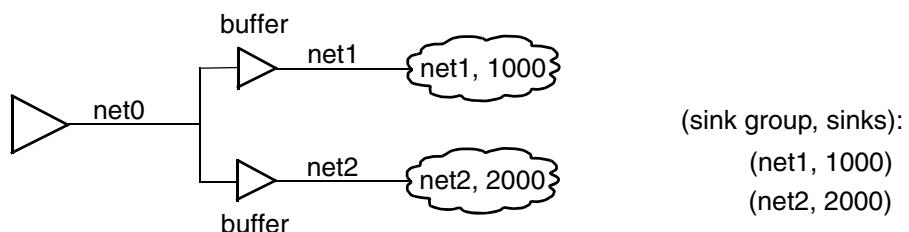
The following restrictions apply:

- When an explicit stop pin or a float pin is not an endpoint of a timing relationship, the tool does not count the timing relationship.
- The tool does not support per-clock exceptions on sinks.

For example, a sink group contains one pin that is defined as a stop pin in one clock domain and as a nonstop pin in another clock domain. When this type of conflict occurs, the tool issues a warning message and excludes this pin from the sink group.

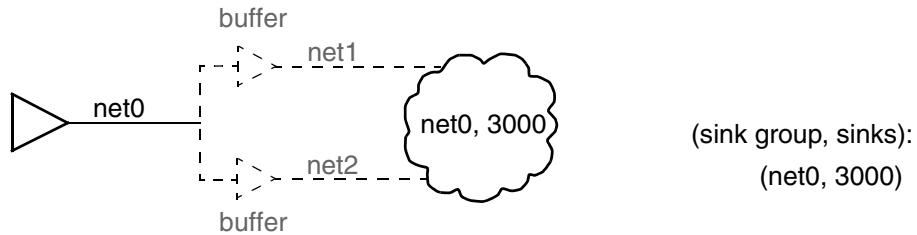
Ignoring Buffers and Inverters

By default, the `report_clock_tree -sink_group` command does not allow the driving nets to pass through preexisting buffers and inverters. As shown in [Figure 5-7](#), the net0 net drives two sink groups, net1 and net2, with a buffer before each sink group. Because the net0 net does not pass through buffers, the resulting sink groups are net1 with 1000 sinks and net2 with 2000 sinks.

Figure 5-7 Example of Clock Sink Groups Driven by Buffers

For the same example shown in [Figure 5-7](#), to enable the `net0` net to pass through preexisting buffers and inverters, specify the `-sink_group_ignore_buf_inv` option with the `report_clock_tree` command. Then, the command treats the buffers as transparent cells and reports the `net0` sink group with 3000 sinks, as shown in [Figure 5-8](#).

Figure 5-8 Example of Passing Through Buffers



The pass-through effect does not occur for a buffer or inverter on which a clock or generated clock has been defined.

Defining the Clock Root Attributes

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 the tool 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), the tool 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]
```

Specifying Clock Tree Exceptions

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, the tool sets PIN_A as a float pin for the CLK1 and CLK2 clock domains.

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

Table 5-3 describes the command options and GUI objects to set the clock tree exceptions supported by the IC Compiler tool.

Table 5-3 Clock Tree Exceptions

Command option	GUI object	Description
Pin exceptions		
-non_stop_pins	“Non stop pins” box	Nonstop pins are pins through which clock tree tracing continues. For more information, see “Specifying Nonstop Pins” on page 5-22 .
-exclude_pins	“Exclude pins” box	Exclude pins are clock tree endpoints that are not used in clock tree timing calculations and optimizations (they are used for design rule constraint calculations and optimizations). For more information, see “Identifying the Clock Tree Endpoints” on page 5-11 and “Specifying Exclude Pins” on page 5-22 .
-float_pins -float_pin_logic_level -float_pin_max_delay_fall -float_pin_max_delay_rise -float_pin_min_delay_fall -float_pin_min_delay_rise	“Float pins” box Max delay fall” box “Max delay rise” box “Min delay fall” box “Min delay rise” box “Logic level” box	Float pins are clock sinks that have special insertion delay requirements. For more information, see “Specifying Float Pins” on page 5-23 .
-stop_pins	“Stop pins” box	Stop pins are clock tree endpoints that are used in calculations and optimizations for both design rule constraints and clock tree timing (skew and insertion delay). For more information, see “Identifying the Clock Tree Endpoints” on page 5-11 and “Specifying Stop Pins” on page 5-27 .

Table 5-3 Clock Tree Exceptions (Continued)

Command option	GUI object	Description
-dont_touch_subtrees	“Don’t touch subtree pins” box	Prevents modification of the clock network beyond the specified pins. The tool considers the sinks of the don’t touch subtree when balancing clock delays or computing the clock skew. For more information, see “ Specifying Don’t Touch Subtrees ” on page 5-27.
Net exceptions		
-dont_buffer_nets	“Don’t buffer nets” box	Prevents buffering of the specified nets during clock tree synthesis. The tool still performs global prerouting on these nets. For more information, see “ Specifying Don’t Buffer Nets ” on page 5-28.
Cell exceptions		
-dont_size_cells	“Don’t size cells” box	Identifies cells that should not be sized during clock tree synthesis and optimization. For more information, see “ Specifying Don’t Size Cells ” on page 5-28.
-size_only_cells	“Size only cells” box	Specifies clock tree cells or instances that can only be sized but not moved during clock tree synthesis and optimization. For more information, see “ Specifying Size-Only Cells ” on page 5-29
-dont_self_gate_registers	“Don’t self-gate” box	Specifies the registers that should not be XOR self-gated. For more information, see “ Using XOR Self-Gating to Save Dynamic Power ” on page 5-30

Table 5-3 Clock Tree Exceptions (Continued)

Command option	GUI object	Description
Hierarchy exceptions		
-preserve_hierarchy	“Preserve hierarchy pins” box	<p>Prevents clustering of clock sinks in the specified local hierarchies that have clock sinks in other logical hierarchies, so the clock pins of the specified logical hierarchies are preserved during clock tree synthesis and optimization.</p> <p>For more information, see “Preserving the Clock Pins of Existing Hierarchies” on page 5-29</p>

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.

If your design contains dangling nets, the clock tree exceptions report might show false implicit ignore pins on these nets. If this happens, you can remove the false implicit ignore pins by saving your design and reopening it.

The following sections provide details for each of the clock tree exceptions.

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. The tool 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 the tool traces through them to find the clock tree endpoints. The clock pins of sequential cells driving generated clocks are implicit nonstop pins. In addition, the tool 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. The tool uses exclude pins only in calculations and optimizations for design rule constraints. In addition to the exclude pins inferred by the tool (the implicit exclude pins), it 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.

During clock tree synthesis, the tool isolates exclude pins (both implicit and explicit) from the clock tree by inserting a guide buffer before the pin. Beyond the exclude pin, the tool 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]
```

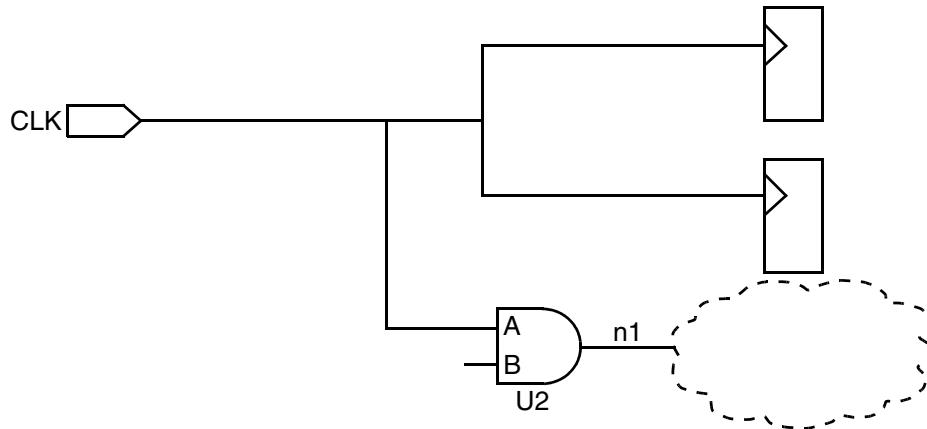
To remove the exclude pin definition from a pin, use the `remove_clock_tree_exceptions -exclude_pins` command.

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]
```

For another example, assume that a clock tree also drives combinational logic, as shown in [Figure 5-9](#).

Figure 5-9 Explicit Exclude Pin



To exclude all branches of the clock tree that fan out from this point, enter

```
icc_shell> set_clock_tree_exceptions -exclude_pins [get_pins U2/A]
```

Specifying Float Pins

Float pins are clock pins that have special insertion delay requirements. The tool 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 5-63.

Note:

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

The IC Compiler tool assumes the active edge of the float pin based on the specified float delay options. [Table 5-4](#) describes the edge-based float pin behavior.

Table 5-4 Edge-Based Float Pin Delays

Options specified	Active edge	Comments
Fall delays only: <code>-float_pin_max_delay_fall</code> <code>-float_pin_min_delay_fall</code>	Falling ¹	If only one of these options is specified, the specified value is used for both the minimum and maximum fall delay.
Rise delays only: <code>-float_pin_max_delay_rise</code> <code>-float_pin_min_delay_rise</code>	Rising ¹	If only one of these options is specified, the specified value is used for both the minimum and maximum rise delay.
Mix of rise and fall delays: <code>-float_pin_max_delay_fall</code> <code>-float_pin_max_delay_rise</code> <code>-float_pin_min_delay_fall</code> <code>-float_pin_min_delay_rise</code>	Both	If one of the fall delay options is not specified, the specified fall delay option is used for both the minimum and maximum fall delays. If one of the rise delay options is not specified, the specified rise delay option is used for both the minimum and maximum rise delays.

1. To define an edge-based stop pin, set the appropriate float pin delay values to zero.

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 in the previous example shows

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

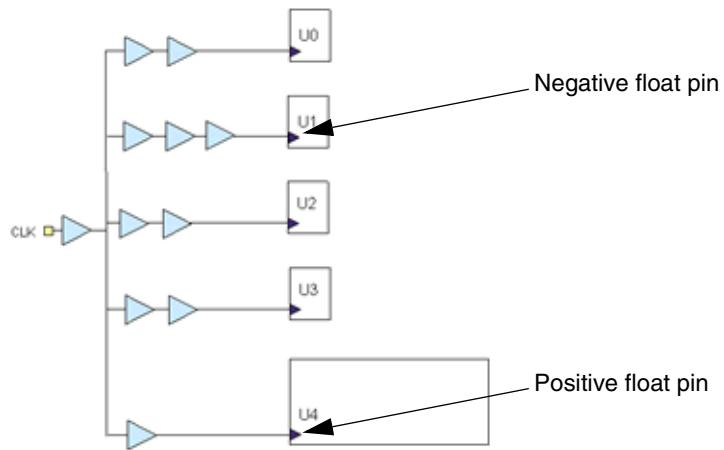
The float pin delay values can be either positive or negative, depending on your timing requirements. To increase the path delay to a pin, specify a negative float pin delay. To decrease the path delay to a pin, specify a positive float pin delay.

For example, the following float pin exceptions result in a clock tree similar to that shown in [Figure 5-10](#):

```
# 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 5-10 Float Pin Timing



For an example of defining float pins, see “[Using Float Pins to Specify Timing for Clock Trees in Hard Macros](#)” on page 5-25.

To remove the float pin definition from a pin, use the `remove_clock_tree_exceptions -float_pins` command.

Using Float Pins to Specify Timing for Clock Trees in Hard Macros

The internal delays for a hard macro cell are represented in the cell’s timing model. The tool 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, the tool 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.

For example, assume that timing analysis shows that the delay of the precompiled clock tree from the RAM_block/CLK port to the earliest sink is 0.33 ns, and the delay to the latest endpoint is 0.52 ns.

The following command defines these timing characteristics for IC Compiler clock tree synthesis:

```
icc_shell> set_clock_tree_exceptions \
    -float_pins [get_pins RAM_block/CLK] \
    -float_pin_max_delay_rise 0.52 -float_pin_max_delay_fall 0.52 \
    -float_pin_min_delay_rise 0.33 -float_pin_min_delay_fall 0.33
```

Specifying Float Pins for Multicorner Clock Tree Optimization

When you specify a float pin value on the leaf node of a clock tree, you must scale the float pin value to apply to each scenario corner. However, achieving a single float pin phase delay across all corners is impossible because cell delays differ in speed. The float pin values that you specify using the `set_clock_tree_exceptions` command apply only to the corners of the default clock tree synthesis scenario in which the clock tree synthesis constraints are specified.

For other corners, specify float pin values by using one of the following two ways:

- Scale float pin values based on the intrinsic speed of cells in different corners by using IC Compiler's automatic capability. For example, if the cells of a corner have a speed twice as fast as that of the default clock tree synthesis scenario, the tool scales down the float pin values that you specify for the default clock tree synthesis scenario by half for the corner.
- Use the `set_clock_tree_exceptions` command with the `-max_float_pin_scale_factor` and the `-min_float_pin_scale_factor` options to specify float pin value scaling factors. You can set the scaling factors on a per scenario basis as shown in the following example:

```
icc_shell> current_scenario scn1
icc_shell> set_clock_tree_exceptions \
    -max_float_pin_scale_factor 1.0 -min_float_pin_scale_factor 0.6

icc_shell> current_scenario scn2
icc_shell> set_clock_tree_exceptions \
    -max_float_pin_scale_factor 0.9 -min_float_pin_scale_factor 0.5
```

Specifying Stop Pins

Stop pins are the endpoints of the clock tree that are used for delay balancing. During clock tree synthesis, the tool 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. In addition, the tool 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.

The tool 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

In some cases you 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 the tool 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`.

The tool 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.

Specifying Don't Buffer Nets

In some cases you might be able to improve the results by preventing the tool from buffering certain nets.

Note:

During clock tree synthesis, the don't buffer net 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]
```

Although clock tree synthesis does not add buffers on such nets, it global routes them.

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, the tool 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

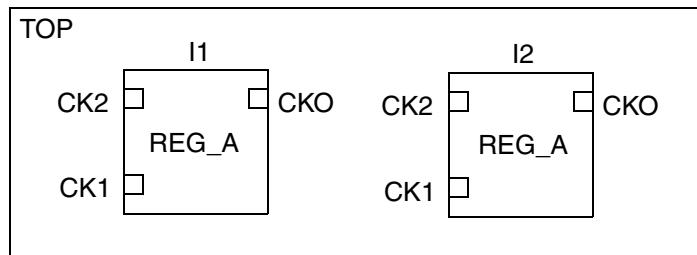
```
icc_shell> remove_clock_tree_exceptions \
-size_only_cells [get_cells U1/U3]
```

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 5-11](#).

Figure 5-11 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, run the following command:

```
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.

Handling Integrated Clock-Gating Cells

When using clock gating to reduce power, you need to consider the impact not only on power and timing, but also on the clock tree QoR. The methodology that you use to insert the integrated clock-gating cells (ICGs) and to optimize the clock tree depends on your power and clock tree QoR goals.

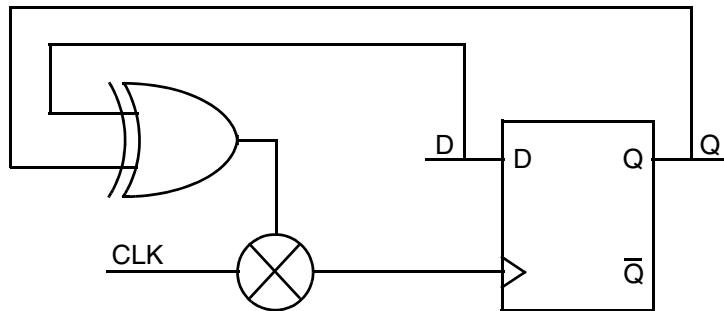
The following sections describe the methodology and techniques:

- [Using XOR Self-Gating to Save Dynamic Power](#)
 - [Optimizing for Clock Tree QoR](#)
 - [Optimizing for Power](#)
 - [Optimizing for Timing on Enable Signals](#)
-

Using XOR Self-Gating to Save Dynamic Power

To reduce the dynamic power consumption of the clock tree network, you can use the advanced clock-gating technique known as XOR self-gating. The XOR self-gating technique turns off the clock signal during the clock cycles when the data in the register remains unchanged. [Figure 5-12](#) shows an example of a register gated by an XOR clock-gating cell. The clock gate is enabled only when switching activity occurs at the register. For more information about XOR self-gating, see the *Power Compiler User Guide*.

Figure 5-12 Example of XOR Self-Gating



To perform XOR self-gating,

1. Ensure the design contains the following:
 - A reference library that contains integrated clock-gating cells and logic gates, including XOR, AND, OR, INV, and NAND, which cannot be marked with the `dont_use` and `dont_touch` attributes.
 - Registers that have a noninverting D input and a Q output to provide the enable signals for the XOR clock-gating cells.
 - A SAIF file that contains the switching activity annotation.
2. Use either of the following two methods:
 - Perform XOR self-gating during standalone clock tree synthesis by specifying the `-insert_self_gating` option with the `compile_clock_tree` command after reading in the SAIF file.

For example,

```
icc_shell> read_saif -input saif_file
icc_shell> compile_clock_tree -insert_self_gating ...
icc_shell> optimize_clock_tree
icc_shell> psynopt
```

- Perform XOR self-gating when implementing the clock trees by specifying the `-insert_self_gating` option with the `clock_opt` command after reading in the SAIF file.

For example,

```
icc_shell> read_saif -input saif_file
icc_shell> clock_opt -insert_self_gating ...
```

To exclude registers from XOR self-gating, specify the `-dont_self_gate_registers register_list` option of the `set_clock_tree_exceptions` command.

To report the power consumption after performing XOR self-gating, use the `report_clock_tree_power` command. To see the power consumption of the registers, specify the `-include_register` option.

To remove all XOR clock-gating cells and the enable logic, use the `remove_self_gating_logic` command. To remove specific XOR clock-gating cells and the corresponding enable logic, specify the `-self_gating_icgs` option with the instance names of the XOR clock-gating cells.

To verify the design in Formality, set the `verification_clock_gate_hold_mode` variable, as shown in the following example:

```
fm_shell> set verification_clock_gate_hold_mode any
```

To avoid design congestion and timing issues later in the flow, you can set the variables that are described in the following section.

Setting the XOR Self-Gating Variables

You can restrict the registers used during XOR self-gating by setting the variables shown in [Table 5-5](#). You should set these variables before running the `compile_clock_tree` or `clock_opt` command.

Table 5-5 XOR Self-Gating Variables

Variable	Default	Description
<code>cts_self_gating_connect_scan_enable</code>	false	Specifies whether to connect the scan enable pin of an XOR self-gated register to one of the corresponding enable nets.
<code>cts_self_gating_data_toggle_rate_filter</code>	0.001	Sets the register data and control pin toggle rate limit so that only those registers with a toggle rate more than this limit are XOR self-gated.
<code>cts_self_gating_num_regs</code>	20000	Sets the maximum number of registers that are XOR self-gated.
<code>cts_self_gating_num_regs_max_percent</code>	24	Sets the maximum percentage of registers (out of the total register count) that are XOR self-gated.
<code>cts_self_gating_num_regs_max_percent_total_cell</code>	7	Sets the maximum percentage of registers (out of the total cell count) that are XOR self-gated.

Table 5-5 XOR Self-Gating Variables (Continued)

Variable	Default	Description
<code>cts_self_gating_reg_power_filter</code>	0.001 mW	Sets the register power threshold such that only registers that use more power than this threshold are XOR self-gated
<code>cts_self_gating_wns_backoff</code>	0.23	Sets a register slack threshold to prevent registers in critical timing paths from being XOR self-gated. Only those registers whose timing slack is better than the threshold are candidates for XOR self-gating. The slack threshold is determined by the following equation: $\text{slack_threshold} = \text{pathgroup_WNS} + (\text{cts_self_gating_wns_backoff_value} * \text{clock_period})$

You can use the `cts_self_gating_wns_backoff` variable to set a register slack threshold based on the worst negative slack (WNS) for the register selection. The following examples use the default of this variable.

For example, consider a path group that has a 1-ns clock period and a WNS of -0.5 ns. The register slack threshold is $-0.5 + (0.23 * 1) = -0.27$; therefore, only registers with slack that is better than -0.27 ns are considered for XOR self-gating.

If the same path group meets the timing constraints, the WNS is 0. The register slack threshold is $0 + (0.23 * 1) = 0.23$, so only registers with slack that is better than 0.23 ns are considered for XOR self-gating.

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:

- Use a small maximum clock-gating fanout value.
- Set the `power_cg_all_registers` variable to `true`.

This variable inserts always-enabled integrated clock-gating cells for ungated registers, so that the clock tree is balanced for both gated and ungated registers.

- Set the `power_remove_redundant_clock_gates` variable to `false`.

This variable prevents Design Compiler from optimizing away the integrated clock-gating cells that are used for balancing.

After placing your design, use the `split_clock_net` command to balance the clock tree fanout before running clock tree synthesis.

Note:

You should use `split_clock_net` to balance the clock tree fanout when the clock trees are unbalanced and the design is not meeting your skew targets, or when you want to relax the constraint on the enable pin of the clock gate by delaying the clock arrival time at the clock gate.

The `split_clock_net` command replicates clock-gating cells to balance the clock tree fanout of the clock gates or clock nets specified in the `-objects` option, which can improve the skew and insertion delay for those clock trees.

```
icc_shell> split_clock_net -objects clk -gate_sizing -gate_relocation
```

The `split_clock_net` command uses the same clustering technique, which is used during clock tree synthesis. This clustering technique improves the skew by using smaller transition time and capacitance constraints, which are based on a technology-dependent computation. This can, however, lead to an increase in clock cell area and power. To force the clustering technique to use a more relaxed constraint, use `set_clock_tree_options` to define the setup values for the constraints and set the `cts_force_user_constraints` variable to `true`. For information about setting the clock tree design rule constraints, see “[Setting Clock Tree Design Rule Constraints](#)” on page 5-50.

The `split_clock_net` command automatically sizes clock gates replicated to minimize the skew in each cluster.

By default, the `split_clock_net` command

- Does not buffer ungated registers

To buffer ungated registers, specify the `-drive_ungated_registers` option when you run the `split_clock_net` command.

- Replicates only integrated clock-gating cells (ICGs)

To replicate any clock gates, specify the `-split_any_cell_type` option when you run the `split_clock_net` command.

- Does not replicate clock gates that have unsynthesized clock subtrees in their fanout

To force replication of these clock gates, specify the `-split_intermediate_level_clock_gates` option when you run the `split_clock_net` command.

The `split_clock_net` command does not replicate the clock-gating cells in the following situations:

- The clock-gating cell has a pin that is defined as a clock (by the `create_clock` command) or as a generated clock (by the `create_generated_clock` command).
- All input pins of the clock-gating cell have a stop pin, exclude pin, or float pin exception.
- The clock-gating cell has a don't buffer net or don't touch subtree exception defined on its input or output pin.
- The clock-gating cell is an integrated clock-gating (ICG) cell inside a hierarchical clock-gating wrapper that also contains other cells.
- The clock-gating cell is inside a block abstraction model.
- All fanout of the clock-gating cell is inside a block abstraction model.

To report the number of clock-gating cells inserted by the `split_clock_net` command, run the `report_clock_tree` command.

Optimizing for Power

When your goal is minimizing power, use a large (or unlimited) maximum clock-gating fanout during insertion of the integrated clock-gating cells (ICGs). After placing your design, perform clock tree power optimization before running clock tree synthesis.

To perform clock tree power optimization,

1. Specify the clock tree references (as described in “[Specifying the Clock Tree References](#)” on page 5-43).
2. Specify the clock tree synthesis options (as described in “[Specifying Clock Tree Synthesis Options](#)” on page 5-45) and exceptions (as described in “[Specifying Clock Tree Exceptions](#)” on page 5-17).
3. Specify the clock tree power optimization settings.
 - To run power-aware placement, set the `-low_power_placement` option to `true` with the `set_optimize_pre_cts_power_options` command.
 - To merge clock-gating cells, set the `-merge_clock_gates` option of the `set_optimize_pre_cts_power_options` command to `true`.
 - To split the clock-gating cells and perform timing optimization on their enable signals, set the `-split_clock_gates` option of the `set_optimize_pre_cts_power_options` command to `true`. To specify the options for splitting the clock-gating cells, use the `set_split_clock_gates_options` command.

When you enable this feature, the tools builds a temporary clock tree during power optimization. Therefore, specify the clock tree synthesis variables, constraints, and exceptions before you perform clock tree power optimization.

For multicorner-multimode designs, you must first set the `set_scenario_options` command options before enabling power optimization.

4. Annotate the switching activity by running the `read_saif` or `set_switching_activity` command.

Important:

If you use a SAIF file to annotate the switching activity, verify that it was generated at the actual clock frequency. If the SAIF file was not generated at the actual clock frequency, the numbers reported for dynamic power might be very small and incorrect.

For more information about annotating the switching activity, see “[Annotating the Switching Activity](#)” on page 3-32.

5. Disable automatic group bounding of the gated clocks by setting the `placer_disable_auto_bound_for_gated_clock` variable to `true`.
6. (Optional) Specify the design constraints used during clock tree power optimization.

You specify the design constraints used during clock tree power optimization in a Tcl script file. You create separate constraint files for clock tree optimization and for post-clock-tree-synthesis optimization.

To specify the name of the constraint file used for clock tree synthesis, use the `-cts_constraint_file` option with the `set_optimize_pre_cts_power_options` command.

To specify the name of the constraint file used for post-clock-tree-synthesis optimization, use the `-psyn_constraint_file` option with the `set_optimize_pre_cts_power_options` command.

7. Perform clock tree power optimization.

To perform standalone clock tree power optimization, run the `optimize_pre_cts_power` command (or choose Clock > Optimize Pre-CTS Power in the GUI). To perform clock tree power optimization as part of the `clock_opt` process, use the `-power` option when you run the `clock_opt` command. For more information about the `clock_opt` command, see “[Implementing Clock Trees](#)” on page 5-84.

Note:

When you run the `clock_opt -power` command, the tool also performs leakage-power optimization. For more information about how to set up for leakage-power optimization, see “[Setting Up for Leakage-Power Optimization](#)” on page 4-36.

The tool performs the following tasks during clock tree power optimization:

1. Merges integrated clock-gating cells that have the same enable signal, if you enable this feature by using the `set_optimize_pre_cts_power_options -merge_clock_gates true` command.
Only integrated clock-gating cells with positive slack are merged; cells with negative slack are not merged.
2. Performs high-fanout net synthesis to fix design rule violations on the enable signals.
3. Performs power-aware placement of the integrated clock-gating cells and registers, if you enable this feature by using the `set_optimize_pre_cts_power_options -low_power_placement true` command.

Note:

If a significant portion of the clock gates are disabled, power-aware placement is not able to effectively optimize the design.

4. Splits the clock gating logic and performs timing optimization on the enable signals of the clock-gating cells, if you enable this feature by using the `set_optimize_pre_cts_power_options -split_clock_gates true` command.

By default, clock tree power optimization ignores `dont_touch` and `size_only` attributes on integrated clock-gating cells. To override these defaults, use the `set_optimize_pre_cts_power_options` command (or choose Clock > Set Optimize Pre-CTS Power Options in the GUI).

To honor `dont_touch` attributes, specify the `-honor_dont_touch` option. To honor `size_only` attributes, specify the `-honor_size_only` option.

To report the current clock tree power optimization options, use the `report_optimize_pre_cts_power_options` command or choose Clock > Report Pre-CTS Power Optimization Options in the GUI.

Optimizing for Timing on Enable Signals

After Power Compiler inserts integrated clock-gating cells (ICGs), timing delays can cause negative slack or timing violations on the enable signals of the integrated clock-gating cells. The timing violations occur because the enable signals are asserted after the clocks become active. To improve the timing on the enable signals, use the `split_clock_gates` command.

Before running the `split_clock_gates` command, you can use the `report_qor` command to check the timing on the enable signals of integrated clock-gating cells. The report shows timing slack based on ideal clocks. If the negative slack of a path is serious, running the `split_clock_gates` command cannot improve the timing. Note that using the

`split_clock_gates` command can optimize timing, but can also increase the area and power.

During the timing optimization step, the `split_clock_gates` command performs the following tasks:

1. Identifies integrated clock-gating cells.

By default, the command finds integrated clock-gating cells in the design and creates a path group of integrated clock-gating cells that is named `icg_en`. If no integrated clock-gating cells are found, the command terminates with a warning message.

2. Runs initial clock tree synthesis.

To include propagated delays on the enable signals of integrated clock-gating cells, the command invokes the `compile_clock_tree` command to build clock trees.

3. Splits integrated clock-gating cells.

When finding an integrated clock-gating cell with negative slack that is smaller than the value specified by the `set_split_clock_gates_options -slack_margin` command, the `split_clock_gates` command invokes the `split_clock_net` command to split the integrated clock-gating cell. If the integrated clock-gating cell is marked with the `is_fixed` attribute or contains a clock definition, the integrated clock-gating cell cannot be split.

4. Removes buffers and inverters.

The command invokes the `remove_clock_tree` command to remove the buffers and inverters that are inserted by the `compile_clock_tree` command during Step 2 so that the design reverts to the unsynthesized stage.

By default, timing optimization ignores the `dont_touch` and `size_only` attributes on integrated clock-gating cells. To override these defaults, use the `set_split_clock_gates_options` command. For example, to honor cells with the `size_only` attribute when running the `split_clock_gates` command, enter

```
icc_shell> set_split_clock_gates_options -honor_size_only  
icc_shell> split_clock_gates
```

To reset the settings, use the `reset_split_clock_gates_options` command. To report the current timing optimization settings, use the `report_split_clock_gates_options` command.

After running the `split_clock_gates` command, use the `report_timing -group icg_en` command to report the timing of integrated clock-gating cells in the design. If running the `split_clock_gates` command does not fix the timing violations on the enable signals, apply the useful skew technique by using the `skew_opt` command. For more information about using the skew technique, see “[Using the Useful Skew Technique](#)” on page 5-157.

Balancing Multiple Clocks

The IC Compiler tool can automatically balance the skew between a group of clocks, either as part of the `clock_opt` process or as a standalone process.

Note:

The tool cannot balance skew between a generated clock and other clocks.

This section describes how to set up the interclock delay balancing requirements. For information about performing interclock delay balancing during the `clock_opt` process, see “[Implementing Clock Trees](#)” on page 5-84. For information about running standalone interclock delay balancing, see “[Running Interclock Delay Balancing](#)” on page 5-103.

The tool also supports interclock delay balancing for multicorner-multimode designs. To set up a multicorner-multimode design, see “[Setting Up for Multicorner-Multimode Analysis and Optimization](#)” on page 3-33.

To define the interclock delay balancing requirements,

1. Define the buffers used for delay balancing.
2. Define the clock balance groups.
3. Define the interclock delay requirements.

Defining the Delay Balancing Buffers

By default, the tool can use all buffers and inverters in your logic library during interclock delay balancing.

To restrict the set of buffers and inverters used during delay balancing, use the `set_clock_tree_references -delay_insertion_only` command. For more information about specifying clock tree references, see “[Specifying the Clock Tree References](#)” on page 5-43.

Defining a Clock Balance Group

By default, the tool balances the delay between all master clocks in the design when you run interclock delay balancing. You can define a clock balance group to restrict the set of clocks considered during delay balancing or to define delay requirements between groups of clocks.

To define a clock balance group, use the `set_inter_clock_delay_options` command (or choose Clock > Set Interclock Delay Options in the GUI). To specify the clock trees in the group, use the `-balance_group` option (or the “Clocks in the group” box in the GUI). You can also assign a name to the balance group by using the `-balance_group_name` option (or the

“Group name” box in the GUI). Although assigning a name to the balance group is optional, you must name the clock group to define a relationship between the group and other clock trees in the design.

For multicorner-multimode designs, the tool balances the interclock delay in a clock balance group while considering the following mode and corners:

- The mode set by the `-cts_mode` option with the `set_scenario_options` command for the scenario in which the balance group is created
- All the corners specified by the `-cts_corner` option with the `set_scenario_options` command

For example, a multicorner-multimode design has the CLK1 and CLK2 clocks defined in the `func_WORST` and `func_BEST` scenarios. When you run the following commands, IC Compiler balances the CLK1 and CLK2 clocks by using the mode from the `func_WORST` scenario and the corners from both the `func_WORST` and `func_BEST` scenarios.

```
icc_shell> current_scenario func_WORST
icc_shell> set_inter_clock_delay_options -balance_group {CLK1 CLK2}
```

If you set the `-cts_mode` option to `true` with the `set_scenario_options` command for a scenario but define no balance groups, the tool automatically creates a default balance group that includes all the clocks.

Defining the Interclock Delay Requirements

By default, the tool has a goal of zero delay offset between clocks. If you have different requirements, you can specify them by using the following methods:

- Specifying a delay offset between clock trees or clock groups
- Using the specified clock latency for each clock

You should not specify multiple interclock delay requirements (for example, specifying both a clock balance group and a delay offset) for a given clock. If a clock has multiple requirements, the tool uses the following precedence when determining the interclock delay requirements:

1. Balancing delays within a clock balance group
2. The delay offset requirements
3. The target insertion delay
4. The SDC clock latency specification, if honored
5. The longest clock network delay

Specifying the Delay Offset Requirements

You specify the delay offset requirement by specifying the following options with the `set_inter_clock_delay_options` command:

- `-delay_offset` (or “Delay offset” box in the GUI)
- `-offset_to` (or “Offset to” box in the GUI)
- `-offset_from` (or “Offset from clock” in the GUI)
- `-offset_from_group` (or “Offset from clock group” box in the GUI)

For example, assume that clk1 has a delay of 500, clk2 has a delay of 700, and clk3 has a delay of 100. If you want to balance the delay between clk1 and clk2 and have clk3 offset by 200, enter the following commands:

```
icc_shell> set_inter_clock_delay_options \
    -balance_group { clk1 clk2 } -balance_group_name grp1
icc_shell> set_inter_clock_delay_options -delay_offset -200 \
    -offset_to clk3 -offset_from_group grp1
```

The result is a delay of 700 for clk1 and clk2, the largest of the delays in the balance group, and a delay of 500 (700-200) for clk3.

As another example, assume that clk1 has a delay of 100 and clk2 has a delay of 200. If you want a delay offset of 200 between clk1 and clk2, enter the following command:

```
icc_shell> set_inter_clock_delay_options -delay_offset 200 \
    -offset_to clk1 -offset_from clk2
```

In this case, the result is a delay of 400 for clk1 (the clk2 delay plus the offset delay) and a delay of 200 for clk2.

Specifying the Target Insertion Delay

To specify the target insertion delay for a clock, specify the clock by using the `-target_delay_clock` option (or “Target delay clocks” in the GUI) and specify the delay value by using the `-target_delay_value` option (or the “Target delay value” box in the GUI).

When you specify a target insertion delay for a clock, the tool increases the insertion delay of the specified clock if it is less than the specified value. If the insertion delay of the specified clock is greater than the target insertion delay, no delay balancing is performed on that clock.

For example, if you want clk1 to have a delay of 100, enter the following command:

```
icc_shell> set_inter_clock_delay_options \
    -target_delay_clock clk1 -target_delay_value 100
```

If the delay on clk1 is less than 100, the tool increases the delay to 100. If the delay on clk1 is greater than 100, the tool does not perform delay balancing on clk1.

For multicorner-multimode designs, the `-delay_offset` and `-target_delay_value` options of the `set_inter_clock_delay_options` command consider the mode and corner from the scenario in which these options are defined.

For example, a multicorner-multimode design has the CLK1 and CLK2 clocks defined in the func_WORST and func_BEST scenarios. When you run the following commands, IC Compiler balances the target delay of 2 ns for the CLK1 clock by using the mode and corner from the func_BEST scenario.

```
icc_shell> current_scenario func_BEST
icc_shell> set_inter_clock_delay_options -target_delay_value 2 \
    -target_delay_clock CLK1
```

Using the Clock Latency Specification

The clock latency is specified by using the `set_clock_latency` command. By default, the tool ignores the clock latency during interclock delay balancing. To honor the SDC clock latency specification, use the `-honor_sdc` option or choose Clock > Interclock Delay Options and select the “Honor latency defined in SDC” check box in the GUI. The `-honor_sdc` option applies to all scenarios for multicorner-multimode designs.

Removing the Interclock Delay Settings

The clock balance groups, target insertion delay, clock latency specification, and the delay offset requirements that are set by the `set_inter_clock_delay_options` command during an IC Compiler session apply to the design in subsequent sessions. To remove the interclock delay settings, use the `reset_inter_clock_delay_options` command with the appropriate options. If you do not specify any options, the command removes all interclock delay settings.

For example, to remove a previously defined clock balance group named grp1, enter

```
icc_shell> reset_inter_clock_delay_options -balance_group_name grp1
```

To remove a list of clock balance groups CLK1 and CLK2, enter

```
icc_shell> reset_inter_clock_delay_options -balance_group "CLK1 CLK2"
```

For example, you use the following command to define interclock delay settings.

```
icc_shell> set_inter_clock_delay_options -offset_from CLK1 \
    -offset_to "CLK2 CLK3" -delay_offset 2
```

To remove the delay offset from CLK1 to both CLK2 and CLK3, enter

```
icc_shell> reset_inter_clock_delay_options -offset_from CLK1
```

To remove the delay offset from CLK1 to CLK2 but to preserve the delay offset from CLK1 to CLK3, enter

```
icc_shell> reset_inter_clock_delay_options -offset_from CLK1 \
           -offset_to CLK2
```

To remove the target insertion delay and clock latency specification, specify the `-target_delay_clock` and `-honor_sdc` options respectively.

To report the interclock delay settings, use the `report_inter_clock_delay_options` command.

Specifying the Clock Tree References

The IC Compiler tool 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 logic 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:

You must ensure that your target library contains at least one inverter that does not have a `dont_use` attribute.

[Table 5-6](#) 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 5-6 Clock Tree Reference List Options

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

When you run the `set_clock_tree_references` command, the tool 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, the tool 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, the tool 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, the tool honors the `dont_use` attribute.

The tool 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}
```

The tool uses this clock tree reference list for all clock trees.

Specifying Clock Tree Synthesis Options

To define clock tree synthesis options, use the `set_clock_tree_options` command (or choose Clock > Set Clock Tree Options in the GUI). [Table 5-7](#) summarizes the clock tree synthesis options supported by the `set_clock_tree_options` command and the corresponding GUI object used to specify them. Some options can be applied either on all clocks (globally) or on a per-clock basis, while other options can be applied globally only.

Table 5-7 Options of the set_clock_tree_options Command

Option	Scope	Default	Description
Clock tree selection options			
<code>-clock_trees</code> ("Clock tree" box)	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 5-50)			
<code>-max_capacitance</code> ("Max capacitance" box in the "Target" tab)	Global or per-clock	0.6 pF	Specifies the maximum capacitance constraint.
<code>-max_fanout</code> ("Max fanout" box in the "Target" tab)	Global or per-clock	2000	Specifies the maximum fanout constraint.
<code>-max_transition</code> ("Max transition" box in the "Target" tab)	Global or per-clock	0.5 ns	Specifies the maximum transition time constraint.
<code>-leaf_max_transition</code> ("Leaf max transition" box in the "Target" tab)	Global or per-clock	Maximum transition setting of the rest of the clock tree	Specifies the maximum transition time constraint for the leaf-level nets.
<code>-use_leaf_max_transition_on_exceptions</code> ("Use leaf max transition on exceptions" box in the "Target" tab)	Global or per-clock	false	Specifies that the maximum transition constraint applied on leaf-level nets by using the <code>-leaf_max_transition</code> option also applies to nets connected to pins with exceptions.

Table 5-7 Options of the set_clock_tree_options Command (Continued)

Option	Scope	Default	Description
-use_leaf_max_transition_on_macros ("Use leaf max transition on exceptions" box in the "Target" tab)	Global or per-clock	false	Specifies that the maximum transition constraint applied on leaf-level nets by using the -leaf_max_transition option also applies to nets connected to macro cells.

Clock tree timing goals (see “Setting Clock Tree Timing Goals” on page 5-53)

-target_skew ("Target skew" box in the "Target" tab)	Global or per-clock	0	Specifies the maximum skew goal.
-target_early_delay ("Target early delay" box in the "Target" tab)	Global or per-clock	0	Specifies the minimum insertion delay goal.

Clock tree routing options (see “Setting Clock Tree Routing Options” on page 5-54)

-layer_list (Layers check boxes in the "Routing" tab)	Global or per-clock	All routing layers	Specifies the preferred layers for clock tree routing.
-layer_list_for_sinks (Layers for sinks check boxes in the "Routing" tab)	Global or per-clock	All routing layers	Specifies the preferred layers for routing nets connected to sinks.
-routing_rule ("Routing rule" box 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" box in the "Routing" tab)	Global	0	Specifies the number of levels starting at the sinks for which to use the default routing, when nondefault routing is specified for clock nets.

Table 5-7 Options of the set_clock_tree_options Command (Continued)

Option	Scope	Default	Description
-routing_rule_for_sinks ("Routing rule for sinks" box in the "Routing" tab)	Global	Default routing rule	Specifies the nondefault routing rule used for nets connected to sinks.
-use_leaf_routing_rule_for_sinks ("Use leaf routing rule for sinks" box in the "Routing" tab)	Global	0	Specifies the number of levels starting at the sinks for which to apply either the nondefault routing rule specified by the -routing_rule_for_sinks option or the preferred layers specified by the-layer_list_for_sinks option.

Boundary cell insertion (see “Inserting Boundary Cells” on page 5-61)

-insert_boundary_cell ("Insert boundary cell" check box in the "Routing" tab)	Global	Disabled	Enables boundary cell insertion.
--	--------	----------	----------------------------------

Clock tree clustering (see “Selecting the Clock Tree Clustering” on page 5-61)

-ocv_clustering ("OCV clustering" check box in the "Routing" tab)	Global	Disabled	Selects clustering based on on-chip variation (OCV).
--	--------	----------	--

Path Sharing (see “Controlling OCV Path Sharing” on page 5-62)

-ocv_path_sharing ("OCV path sharing" check box in the "Routing" tab)	Global	Enabled	Enables OCV clock path sharing.
--	--------	---------	---------------------------------

Table 5-7 Options of the set_clock_tree_options Command (Continued)

Option	Scope	Default	Description
Logic-level balancing (see “ Enabling Logic-Level Balancing ” on page 5-63)			
-logic_level_balance (“Logic level balance” check box in the “Routing” tab)	Global	Disabled	Enables logic-level balancing.
Clock tree DRC violation fixing (see “ Fixing DRC Violations on Clock Nets ” on page 5-71)			
-advanced_drc_fixing (“Advanced DRC fixing” check box in the “Routing” tab)	Global	Disabled	Enables advanced DRC fixing techniques.
Embedded clock tree optimizations (see “ Specifying Clock Tree Optimization Options ” on page 5-69)			
-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. ¹
-gate_sizing (“Gate sizing” check box)	Global or per-clock	Disabled	Enables gate sizing during optimization.

Table 5-7 Options of the set_clock_tree_options Command (Continued)

Option	Scope	Default	Description
Clock tree configuration file options (see “ Inserting User-Specified Clock Trees ” on page 5-73)			
-config_file_read (“Read file” box in “Configuration files” box)	Global	None	Specifies the clock configuration file to read.
-config_file_write (“Write file” box in “Configuration files” box)	Global	None	Specifies the name of the clock configuration file that is written after clock tree synthesis.

1. Cells with the `is_fixed` attribute are not moved during 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 the clock tree synthesis options to their defaults, 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

The IC Compiler tool 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

The IC Compiler tool 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).

Note:

The tool does not support the specification of per-clock design rule constraints for overlapping clock domains.

Having a short transition time at the clock pin of registers helps improve the timing QoR of the design. To achieve a short transition time, you can set a tight maximum transition constraint for the entire clock tree. However, this can increase the number of buffers used to build the clock tree. You can set a maximum transition constraint specifically for the leaf-level nets by using the `-leaf_max_transition` option with the `set_clock_tree_options` command.

When you use the `-leaf_max_transition` option, you can apply the same constraint for nets connected to pins of macro cells by using the `-use_leaf_max_transition_on_macros` option. You can also apply the same constraint for nets connected to pins with float-pin or stop-pin exceptions by using the `-use_leaf_max_transition_on_exceptions` option.

In addition to the clock tree design rule constraint values that you specify, the tool also considers the design rule constraint values from the logic library and the design. [Table 5-8](#) summarizes how the tool 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).

Table 5-8 Clock Tree Synthesis Design Rule Constraints

Variable settings			
Design rule constraint	Default behavior: <code>cts_use_lib_max_fanout =false</code> <code>cts_use_sdc_max_fanout =false</code> <code>cts_force_user_constraints=false</code>	Use library and SDC settings for maximum fanout: <code>cts_use_lib_max_fanout =true</code> <code>cts_use_sdc_max_fanout =true</code> <code>cts_force_user_constraints=false</code>	Use only clock tree synthesis settings for clock tree synthesis and clock tree optimization: <code>cts_force_user_constraints=true</code>
Maximum capacitance	The minimum value from <ul style="list-style-type: none">• The clock tree synthesis value• The logic library• The SDC constraints	The minimum value from <ul style="list-style-type: none">• The clock tree synthesis value• The logic library• The SDC constraints	The clock tree synthesis value
Maximum transition time	The minimum value from <ul style="list-style-type: none">• The clock tree synthesis value• The logic library• The SDC constraints	The minimum value from <ul style="list-style-type: none">• 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 <ul style="list-style-type: none">• The logic library• The SDC constraints	The clock tree synthesis value

To see the design rule constraint settings that are used for clock tree synthesis, generate the clock tree settings reports, as described in “[Generating Clock Tree Reports](#)” on page 5-135. Review these values. If any of the values are incorrect for clock tree synthesis, you can override them by setting clock tree design rule constraints.

Note:

The clock tree settings report displays the design rule constraints based on the default behavior; it does not consider the effects of the `cts_force_user_constraints` or `cts_use_lib_max_fanout` variables.

Setting the Maximum Transition Time Constraints

You can set the maximum transition time constraints to datapaths and clock paths by using the `set_max_transition` command. By default, the maximum transition time constraints apply to all pins on the datapaths and clock paths launched by the specified clocks for the current scenario only. To restrict the constraints to datapaths only, specify the `-data_path` option. To restrict the constraints to clock paths only, specify the `-clock_path` option.

Alternatively, you can use the `set_clock_tree_options` command with the `-max_transition` option to constrain clock paths.

For example, the following command applies a maximum transition time of 0.200 ns to the `clk1` clock path:

```
icc_shell> set_max_transition 0.200 -clock_path [get_clocks clk1]
```

When an SDC file contains multiple maximum transition time constraints set by the `set_max_transition` and `set_clock_tree_options -max_transition` commands for the same clock path, the strictest constraint value is honored. As shown in the following example, an SDC file contains three maximum transition time constraints for the `clk1` clock path; the tool applies the maximum transition time of 0.1 ns to the `clk1` clock path.

```
icc_shell> set_max_transition 0.1 [current_design]
icc_shell> set_max_transition 0.2 [get_clocks clk1]
icc_shell> set_max_transition 0.3 -clock_path [get_clocks clk1]
```

However, when you set the `cts_force_user_constraints` variable to `true`, only the constraints set by the `set_clock_tree_options -max_transition` command are honored.

The following two criteria show the behavior differences between the IC Compiler and PrimeTime tools when you run the `set_max_transition` command:

- A maximum transition time constraint is specified on a master clock, but no constraint is set on its generated clock.

The generated clock inherits the stricter value of the maximum transition time constraints set on the current design and on the master clock. In PrimeTime, the generated clock inherits the maximum transition time constraint that is set on the current design. For example, when you run the following commands, the resulting maximum transition time for the generated clock in the `clock1` master clock domain is 0.3 ns in IC Compiler but 0.5 ns in PrimeTime.

```
icc_shell> set_max_transition 0.5 [current_design]
icc_shell> set_max_transition 0.3 -clock_path [get_clock clock1]
```

- Both the `-clock_path` and `-data_path` options are specified for the same path.

Clock tree synthesis and optimization honors the setting of the `-clock_path` option, whereas PrimeTime honors the stricter setting of the two options.

Setting Clock Tree Timing Goals

During clock tree synthesis, the tool 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).

Note:

The tool can consider the clock latency specification during interclock delay balancing.
For more information, see “[Balancing Multiple Clocks](#)” on page 5-39.

You can specify the following clock tree timing goals for a clock tree:

- Maximum skew (`set_clock_tree_options -target_skew`)

During optimization, the tool computes the skew value by comparing the arrival times of all clock signals in a clock domain, including those that do not communicate through datapaths (global skew).

If you do not specify a maximum skew value, the tool uses zero as the maximum skew.

- Minimum insertion delay (`set_clock_tree_options -target_early_delay`)

The tool 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, the tool 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 tree timing goals for all clocks, omit the `-clock_trees` option.

For multicorner-multimode designs, you specify the same skew value for all corners, as shown in following example:

```
icc_shell> set_clock_tree_options -target_skew 0.050
```

To specify various skew values for different corners, use the `set_clock_tree_optimization_options` command, as shown in the following example:

```
icc_shell> set_clock_tree_optimization_options \
    -corner_target_skew "scn1:max=0.100 scn_cts:max=0.050 scn2:min=0.070"
```

Setting Clock Tree Routing Options

The IC Compiler tool 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 pin, port, or cell type
- Whether to honor existing detail routes

Specifying Routing Rules

If you do not specify which routing rule to use for clock tree synthesis, the tool 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 predefined `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, nondefault routing rules are often not required on the nets closest to the clock sinks. To use default wires on the nets connected to the clock sinks, use the `-use_default_routing_for_sinks` option and specify the number of levels of nets starting at the sinks for which you want to use default routing.

To specify a nondefault routing rule for the nets connected to sinks, use the `-routing_rule_for_sinks` option. When you use this option, you must also specify the number of levels starting at the sinks for which you want to use the nondefault routing by using the `-use_leaf_routing_rule_for_sinks` option.

For example, to use the `clk_leaf_rule` routing rule for one level of nets starting at the sinks, use the following command:

```
icc_shell> set_clock_tree_options -routing_rule_for_sinks clk_leaf_rule \
    -use_leaf_routing_rule_for_sinks 1
```

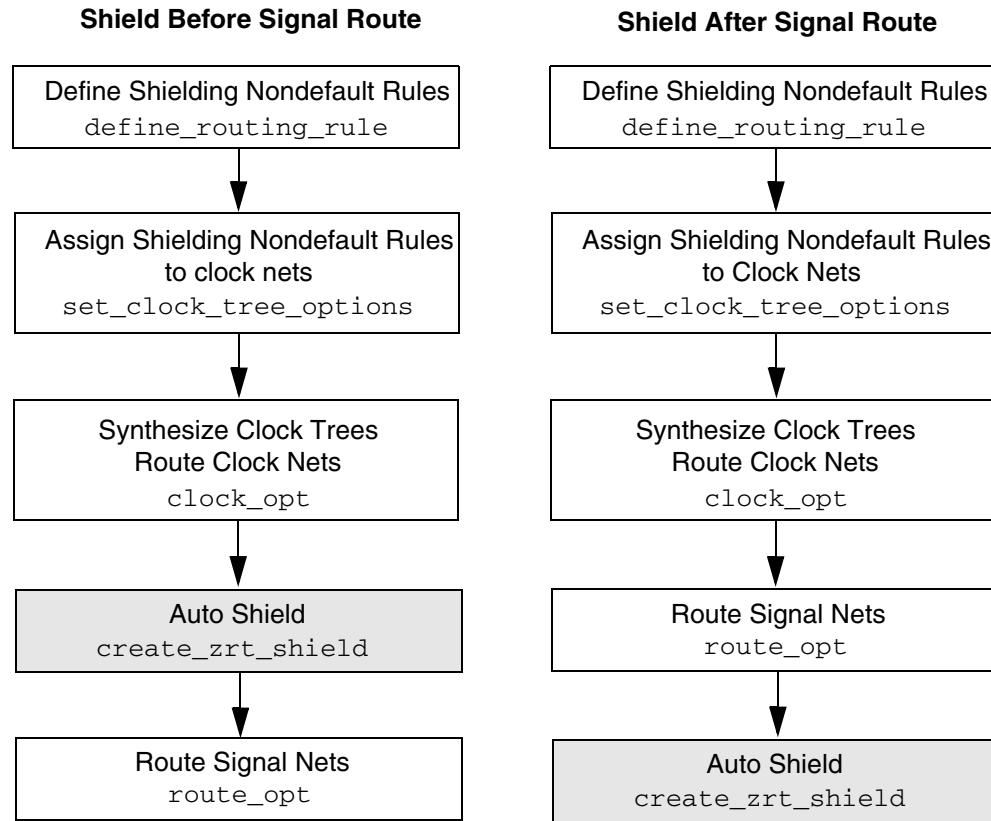
To see the nondefault routing rules defined for the clock trees in your design, run the `report_clock_tree -settings` command.

For finer control of the clock tree routing rules, you can specify the routing rules in a clock configuration file. For information about the clock configuration file, see “[Inserting User-Specified Clock Trees](#)” on page 5-73. Clock tree routing rules defined in a clock configuration file override those set by using the `set_clock_tree_options` command.

Shielding Clock Nets

The IC Compiler tool 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 5-13](#) shows the flow of both clock shielding methodologies.

Figure 5-13 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 6-16.

The syntax to assign a nondefault routing rule to unsynthesized clock trees is

```
set_clock_tree_options
    [-clock_trees clock_tree_names]
    [-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 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_trees 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
```

For detailed information about shielding, see “[Shielding Nets](#)” on page 7-41.

Specifying Routing Layers

If you do not specify which routing layers to use for clock tree synthesis, the tool 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, to specify that the clock nets for clock CLK1 should be routed on layers M4 and M5, use the following command:

```
icc_shell> set_clock_tree_options -clock_trees CLK1 \
           -layer_list {M4 M5}
```

When you specify the clock tree routing layers by using the `-layer_list` option, the specified layers apply to all levels of the clock tree. However, you can specify preferred routing layers for only the nets connected to sinks by using the `-layer_list_for_sinks` option. When you use this option, you must also specify the number of levels starting at the sinks for which you want to use the preferred routing layers by using the `-use_leaf_routing_rule_for_sinks` option.

For example, to specify that one level of nets starting at the sinks should be routed on layers M3 and M4, use the following command:

```
icc_shell> set_clock_tree_options -layer_list_for_sinks {M3 M4} \
           -use_leaf_routing_rule_for_sinks 1
```

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 5-73. Clock tree layer constraints defined in a clock configuration file override those set by using `set_clock_tree_options`.

By default, the tool 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 -net_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.

Specifying Routing Rules and Layers for Instance Pins and Ports

To specify a routing rule and a list of layers to use for a clock net that is driven by a specific instance pin or port, use the `set_instance_based_routing_rule` command.

The following example shows how to specify the CLK_NDR routing rule and the M5 and M6 layers for the clock net driven by the CLK_20/Z pin:

```
icc_shell> set_instance_based_routing_rule CLK_20/Z \
    -routing_rule CLK_NDR -layer_list {M5 M6}
```

The tool uses the routing rule and layer list settings when you perform clock tree synthesis and optimization by using the `clock_opt`, `compile_clock_tree`, or `optimize_clock_tree` command.

To report these settings, use the `report_instance_based_routing_rule` command. To remove these settings, use the `reset_instance_based_routing_rule` command.

Specifying Routing Rules for 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 specify compatible nondefault routing rules for reference cells by using the `set_reference_cell_routing_rule` command.

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 specify nondefault routing rules for reference cells,

- If multiple nondefault routing rules are specified for 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:

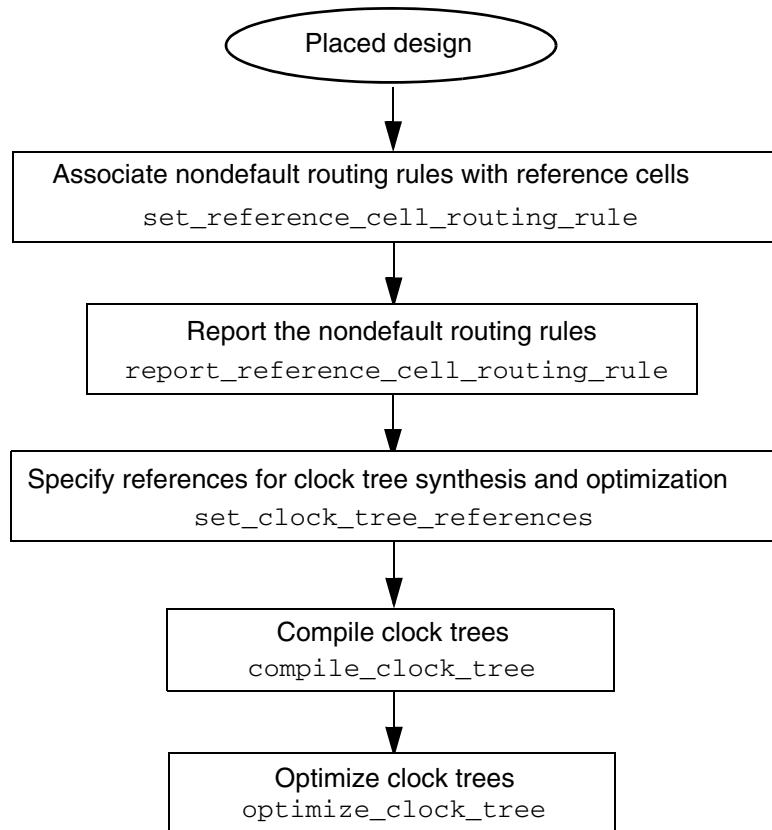
The `-use_default_routing_for_sinks` option of the `set_clock_tree_options` command is not supported with these nondefault routing rules. If you use both the `-use_default_routing_for_sinks` option and nondefault routing rules associated with reference cells, the `-use_default_routing_for_sinks` option 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 5-14](#) shows the flow of associating nondefault routing rules with reference cells.

Figure 5-14 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 5-43.)
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, the tool disables the boundary cell insertion and generates a warning message.

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

The tool 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, the tool 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

The IC Compiler tool performs clustering of the clock sinks to minimize wire length. If your design is sensitive to on-chip variation, the tool 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 5-73.

Enabling OCV Clustering

The optional OCV clustering algorithm considers the timing constraints between clock sinks to influence clustering. Sinks with timing-critical paths driven by the same gates are clustered together. To enable OCV 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 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 clustering can increase the runtime and power.

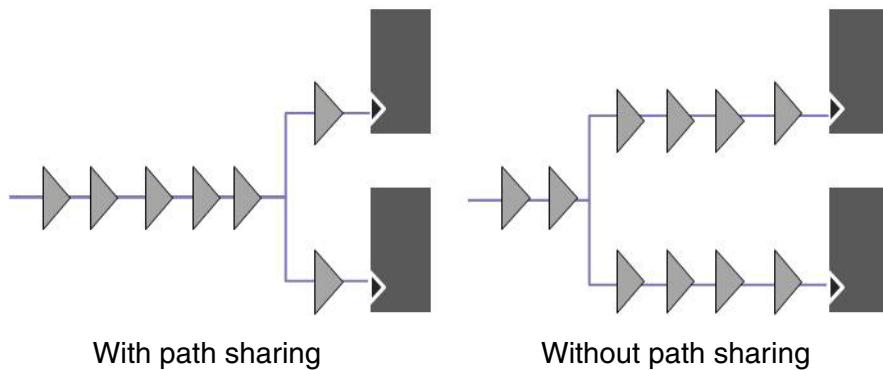
Note:

You cannot use OCV clustering with clock tree configuration files. If you specify the `-ocv_clustering` option when these options are set, the tool ignores the `-ocv_clustering` option. If you specify a clock tree configuration file after setting the `-ocv_clustering` command, the tool issues an error message and ignores both settings.

Controlling OCV Path Sharing

For designs with OCV derating factors, the tool can reduce the clock skew caused by on-chip variation by increasing the sharing of clock paths that lead to different sinks within the same clock group. When you increase path sharing, the local skew for each clock group and the timing QoR for paths within each clock group improves; however, this feature has no effect on timing paths that span between clocks groups. [Figure 5-15](#) shows a clock path with and without path sharing.

Figure 5-15 With and Without Clock Path Sharing



During clock tree synthesis, the tool performs OCV path sharing by default. However, if the difference between the early and late timing derating factors set on the clock nets is less than 0.09 for all the scenarios enabled for clock tree synthesis, the tool does not perform OCV path sharing as the benefits are negligible.

For example, if you specify the following timing derating factors, the tool does not perform OCV path sharing because the difference between the early and late timing derating factors on the clock nets is 0.04:

```
icc_shell> set_timing_derate -clock -early 0.98  
icc_shell> set_timing_derate -clock -late 1.02
```

You can change the threshold that is used to disable OCV path sharing by using the `cts_ocv_path_sharing_derating_threshold` variable. Set this variable to zero to ensure that the OCV path sharing is always enabled.

You can disable OCV path sharing by setting the `-ocv_path_sharing` option of the `set_clock_tree_options` command to `false`. Setting it to `false` affects all clock trees in the design, regardless of the setting for the `-clock_trees` option of the `set_clock_tree_options` command.

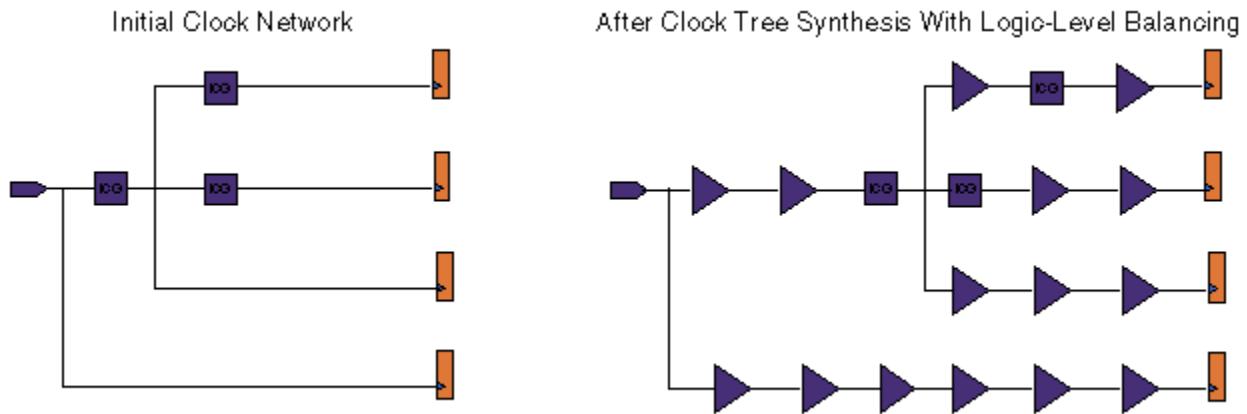
Enabling Logic-Level Balancing

If on-chip variation is an issue for your design, use the logic-level balancing mode.

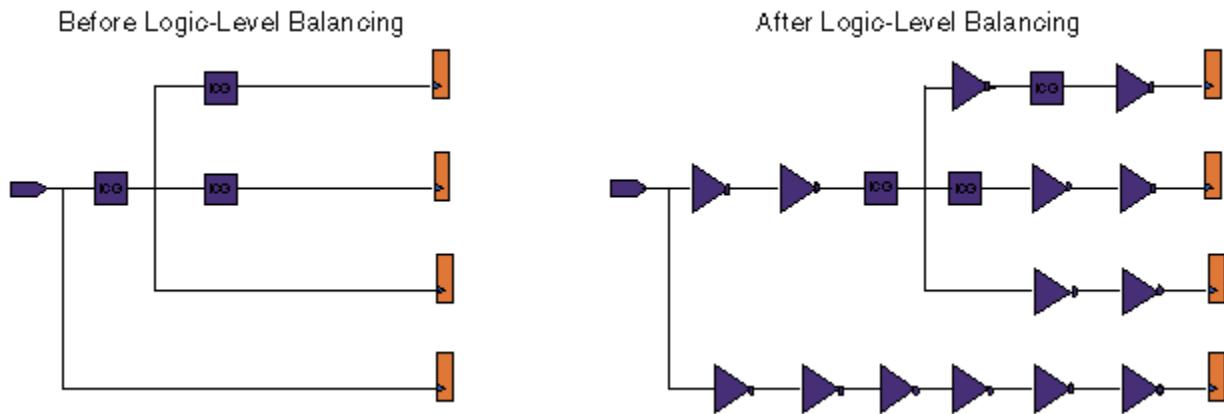
Note:

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, the tool balances the delay in each branch of the clock tree, but it does not consider the number of logic levels in each branch. The tool 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 5-16](#) shows a clock tree that was synthesized using logic-level balancing.

Figure 5-16 Logic-Level Balancing

Logic-level balancing can use buffers, inverters, or both to balance the logic levels. If you use only inverters for logic-level balancing and the initial clock network does not have balanced logic levels, the generated clock trees might be unbalanced by one level, as shown in [Figure 5-17](#).

Figure 5-17 Logic-Level Balancing Using Inverters

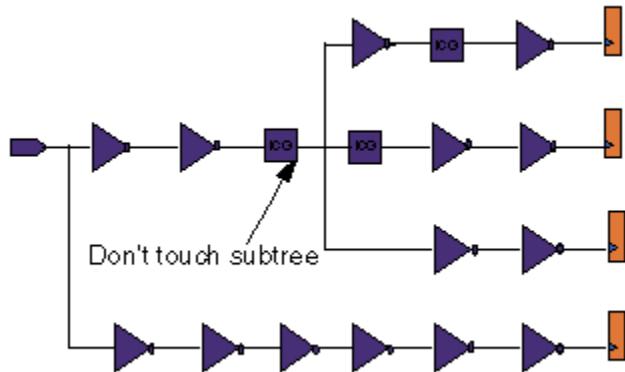
To enable logic-level balancing, use the `set_clock_tree_options -logic_level_balance true` command.

If you enable logic-level balancing, it applies to all clock trees. You cannot enable logic-level balancing on a per-clock basis.

If a clock tree contains a subtree that is not modified during clock tree synthesis (either a don't touch subtree or a subtree within a block abstraction model), the tool traces through the subtree to determine the number of logic levels contained in the subtree. The tool considers these logic levels when constructing the clock tree. If the subtree does not have

balanced logic levels, the tool generates a warning message and uses the maximum number of levels in the subtree as the number of logic levels for the subtree. For example, for the don't touch subtree shown in [Figure 5-18](#), the tool uses four as the number of logic levels in the don't touch subtree.

Figure 5-18 Logic-Level Balancing With a Don't Touch Subtree



If your design contains hard macros, use the `set_clock_tree_exceptions -float_pin_logic_level` command to specify the number of logic levels within the hard macro. The number of logic levels must be a positive integer. If you do not specify the number of logic levels and the tool cannot derive the number of logic levels, the tool assumes that there are no logic levels within the hard macro.

Note:

When you remove float pin logic-level information by using the `remove_clock_tree_exceptions -float_pin_logic_level` command, the tool removes logic-level information for the entire design. It does not remove logic-level information for an individual pin.

After clock tree synthesis finishes, the tool verifies that the clock trees are balanced. If the number of logic levels varies between branches, the tool generates a warning message.

To enable OCV clustering while running logic-level balancing, set both the `-logic_level_balance` and `-ocv_clustering` options to `true` with the `set_clock_tree_options` command. For more information about OCV clustering, see [“Enabling OCV Clustering” on page 5-62](#).

Warning:

If you use logic-level balancing, do not run clock tree optimization with delay insertion. Doing so can unbalance the logic levels. If you use logic-level balancing when running the `clock_opt` command, delay insertion is automatically disabled during the embedded clock tree optimization.

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 exclusive move bounds

Note:

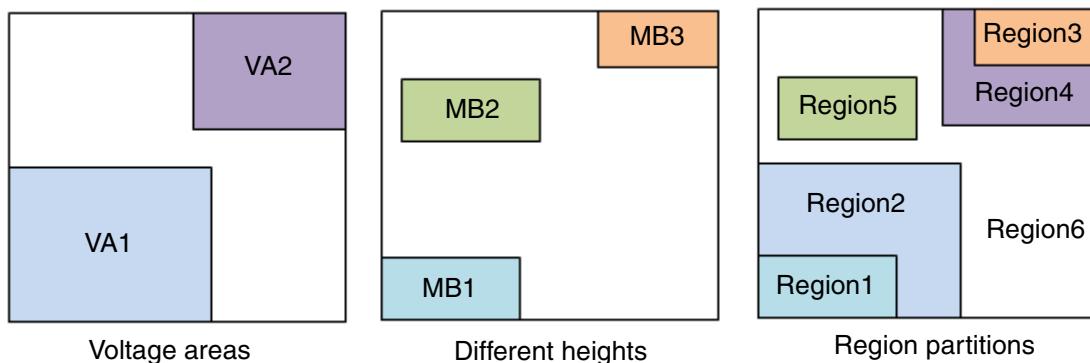
Soft move bounds and hard move bounds are ignored during clock tree synthesis.

- 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:

- 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 exclusive move bounds.
- Partitions a design into regions according to constraints such as voltage areas, plan groups, and exclusive move bounds, as shown in [Figure 5-19](#).
- Associates each buffer group with a region and vice versa.
- Associates each power guide with the region that contains it.
- Performs region-aware clustering.

Figure 5-19 Design Partitions



Reducing Electromigration

Clock cells consume more power than cells that are not in the clock network. Clock cells that are clustered together in a small area increase current densities for the power and ground rails, where a potential electromigration problem might occur. One way to avoid the problem is to set spacing requirements between clock cells. You set the spacing requirements by defining clock cell spacing rules for inverters, buffers, and integrated clock-gating cells in the clock network. Thus, you can prevent local clumping of the clock cells along a standard cell power rail between the perpendicular straps. You should not attempt to globally space the clock cells on the design because it can result in an area requirement larger than the size of the clock tree synthesis clusters of sinks.

To define clock cell spacing rules, use the `set_clock_cell_spacing` command:

- Set either or both of the `-x_spacing` and `-y_spacing` options to a nonzero value. The `-x_spacing` value is usually dependent on the library cell's drive strength and clock frequency. The `-y_spacing` value is usually less than half a row height. Each setting specifies the minimum distance, in microns, from the adjacent clock cells in the specified direction. If two adjoining clock cells both have a spacing value specified in a given direction, the sum of the spacing values applies. For example, if the `-x_spacing` option value is 5 for two adjoining clock cells, they are placed at least 10 microns apart in the `x`-direction.

You can relax the adjoining cell spacing rule by setting the `legalize_use_cts_max_spacing` variable. When this variable is set to `true`, IC Compiler uses the larger of the spacing settings between two adjoining cells instead of the sum of the spacing settings. For example, two adjoining clock cells, each with a `-x_spacing` option value of 5, are placed at least 5 microns apart in the `x`-direction.

Note:

Using a large spacing value increases the skew because the cells are spaced further away from the intended clusters of sinks.

- Optionally restrict the clock cell spacing rules to a collection of library cells by using the `-lib_cells` option or to a collection of clock names by using the `-clocks` option.

To report clock cell spacing rules, use the `report_clock_cell_spacing` command. The report categorizes information into three sections:

1. Clock cell spacing rules set by the `-lib_cells` option only or by no option.
2. Clock cell spacing rules set by the `-clocks` option only.
3. Clock cell spacing rules set by both the `-lib_cells` and `-clocks` options.

To remove clock cell spacing rules, use the `remove_clock_cell_spacing` command. You can specify the `-lib_cells` and `-clocks` options to remove clock cell spacing constraints from the specified library cells and clocks respectively.

If you use the `remove_clock_cell_spacing` command with

- The `-lib_cells` option

The command removes only the clock cell spacing rules defined by the `set_clock_cell_spacing -lib_cells` command with the specified library cells.

- The `-clocks` option

The command removes only the clock cell spacing rules defined by the `set_clock_cell_spacing -clocks` command with the specified clock names.

- Both the `-lib_cells` and `-clocks` options

The command removes the clock cell spacing rules defined by the `set_clock_cell_spacing -lib_cells -clocks` command with the specified library cells and clock names.

- No option

The command removes the clock cell spacing rules defined by the `set_clock_cell_spacing` command with no option.

You use the following steps to reduce electromigration in the design:

1. After obtaining a placement CEL view, set the clock cell spacing rules by using the `set_clock_cell_spacing` command.

To remove and report the clock cell spacing rules, use the `remove_clock_cell_spacing` and `report_clock_cell_spacing` commands respectively.

2. Perform clock tree synthesis by using either the `compile_clock_tree` or `optimize_clock_tree` command.

3. Check clock cell spacing rule violations by using the `check_legality -verbose` command.

You should not see any violations if you set the appropriate clock cell spacing constraints.

Note that the `compile_clock_tree`, `optimize_clock_tree`, `split_clock_net`, and `balance_inter_clock_delay` commands support clock cell spacing rules.

Because clock cell spacing is a preventive and not a corrective technique, the `optimize_clock_tree` command does not correct the spacing violations that remain unfixed after the `compile_clock_tree` command. Clock tree optimization or the `psynopt` command does not fix spacing violations caused by changing spacing rules in the middle of the flow.

Enabling Arnoldi Delay Calculation

To enable Arnoldi delay calculation for clock tree synthesis, use the following variable setting:

```
icc_shell> set_app_var cts_use_arnoldi_based_delays true
```

Using Arnoldi delays during clock tree synthesis helps improve the timing correlation with the subsequent steps in the design flow. However, it can increase the runtime. Use this feature for technology nodes of 20 nm or below. For technology nodes above 20 nm, the possible runtime increase might outweigh the correlation benefit.

Specifying Clock Tree Optimization Options

The IC Compiler tool optimizes the clock trees during the design stages that are shown in [Table 5-9](#). During the optimization phases, the tool can perform several optimization tasks, which you can enable or disable by setting the appropriate options.

Table 5-9 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:

The tool 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 5-50](#).

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 5-10](#) shows the available options and their defaults.

Table 5-10 Embedded Clock Tree Optimization Options

Option	Default	Description
<code>-buffer_relocation</code>	true	Optimizes the placement of the buffers and inverters in the synthesized clock trees.
<code>-buffer_sizing</code>	true	Optimizes the sizing of the buffers and inverters in the synthesized clock trees.
<code>-gate_relocation</code>	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.
<code>-gate_sizing</code>	false	Optimizes the sizing of the preexisting gates in the clock trees.

Controlling Preroute Clock Tree Optimization

During `clock_opt`, the clock tree optimization phase performs all the optimization tasks listed in [Table 5-11](#). 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 5-11](#) shows the available options and their defaults.

Table 5-11 Preroute Clock Tree Optimization Options

Option	Default	Description
<code>-buffer_relocation</code>	true	Optimizes the placement of the buffers and inverters in the synthesized clock trees.
<code>-buffer_sizing</code>	true	Optimizes the sizing of the buffers and inverters in the synthesized clock trees.

Table 5-11 Preroute Clock Tree Optimization Options (Continued)

Option	Default	Description
-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 5-12** shows the available options and their defaults.

Table 5-12 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.

Fixing DRC Violations on Clock Nets

To ensure that all DRC violations on clock nets are fixed during clock tree synthesis and optimization, enable advanced DRC fixing techniques by using the `set_clock_tree_options -advanced_drc_fixing true` command before you run the `compile_clock_tree`, `clock_opt`, or `optimize_clock_tree` command.

Advanced DRC fixing is disabled by default and enabling it affects all clocks regardless of the clock you specify with the `-clock_trees` option of the `set_clock_tree_options` command.

Fixing DRC Violations on Signal Nets

Clock tree synthesis considers some nets as signal nets, such as a net in the clock network that is connected to the data input of a flip-flop or to an output port. Clock tree synthesis does not fix DRC violations on the signal nets. By default, the `psynopt` command fixes DRC violations on the signal nets after clock tree synthesis. To enable automatic DRC fixing on the signal nets during clock tree synthesis, set the `cts_enable_drc_fixing_on_data` variable to `true`, changing it from its default of `false`.

When this variable is set to `true`,

- The `compile_clock_tree` and `clock_opt` commands fix DRC violations on the signal nets in the clock network based on the constraints of clock tree synthesis.
 - The `remove_clock_tree` and `mark_clock_tree` commands also apply to the signal nets in the clock network.
 - The `report_clock_tree -drc_violators` command includes DRC violations on the signal nets in the clock network.
-

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 4-45.

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

The IC Compiler tool 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 5-75.

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, the tool 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 the tool 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, the tool 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, the tool 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, the tool determines the routing rule as follows:

- If the clock tree structure is completely specified, the tool 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, the tool 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 5-54](#).

`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, the tool 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.

For more information about nondefault routing rules, see “[Specifying Routing Layers](#)” on [page 5-58](#).

`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, the tool 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, the tool 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, the tool 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 the tool 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 the tool 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
```

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, the tool keeps an existing clock tree and might modify it. Instead, you can either prevent the tool 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” box 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, the tool uses the clock roots defined by the `create_clock` and `create_generated_clock` commands.

The tool traverses the clock tree from the specified startpoint and sets the clock tree attributes as if the tree were synthesized. Clock tree traversal continues until it finds an exception pin or a default sink pin.

After running `mark_clock_tree -clock_synthesized`, the imported clock trees are treated just like clock trees synthesized by IC Compiler: the buffers and inverters are fixed; the clock sink cells can only be sized, not moved; the clocks are set to propagated; and the user-defined clock latency values are removed. Similar to using the `cts_fix_clock_tree_sinks` variable to fix clock sink cells after clock tree synthesis, you can use the `-fix_sinks` option when you run `mark_clock_tree` to prevent any modification to the clock sink cells after clock tree synthesis. For more information about the IC Compiler clock tree synthesis process, see “[Implementing Clock Trees](#)” on page 5-84.

Important:

Because the buffers and inverters (and possible clock sink cells) are fixed after running `mark_clock_tree`, you must ensure that the placement is legal before running the `mark_clock_tree` command.

Preserving Portions of an Existing Clock Tree

In some cases you 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. For information about don't touch subtrees, see “[Specifying Don't Touch Subtrees](#)” on page 5-27.

Removing Clock Trees

To remove a clock tree, use the `remove_clock_tree` command (or choose Clock > Remove Clock Tree in the GUI).

When the tool 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 the tool (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.

[Table 5-13](#) shows how clock tree removal is affected by the structure of the clock tree.

Table 5-13 Clock Tree Removal Behavior

Object	Impact on clock tree removal
Boundary cells	Removed.
Cells on don't buffer nets	Preserved.
Don't touch subtree	Preserved.
Fixed cells	Preserved.
Size-only cells	Preserved.

Table 5-13 Clock Tree Removal Behavior (Continued)

Object	Impact on clock tree removal
Generated clock	Preserved, if generated clock is defined on buffer/inverter pin. Traversal and clock tree removal continue past the generated clock.
Guide buffer	Removed.
Integrated clock-gating (ICG) cell	Preserved. Traversal (and clock tree removal) continues past the integrated clock-gating cell.
Block abstraction	Preserved. Traversal (and clock tree removal) continues past the block abstraction.
Inverter	Removed in pairs only. If a clock tree contains a single inverter, it is not removed.
Isolation cell	Preserved.
Level shifter	Preserved.
Three-state buffer	Preserved. Traversal (and clock tree removal) stops at the three-state buffer.
Buffer or inverter added beyond exclude pin, stop pin, or float pin	Removed.

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
```

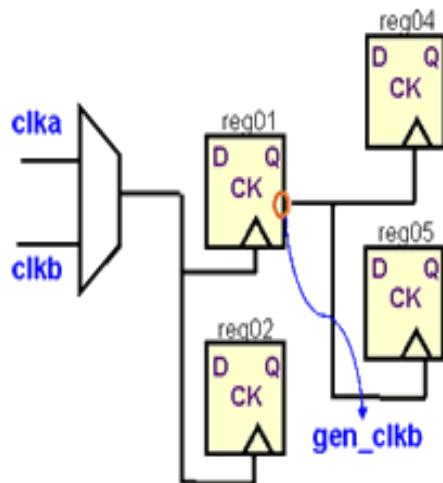
If you do not specify the `-clocks` option, the tool checks all clocks in the current design.

The `check_clock_tree` command checks for the following issues:

- 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 5-20](#), 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 5-20 Generated Clock With Invalid Master Clock Source



- Clock (master or generated) with no sinks
- Looping clock
- 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

For multicorner-multimode designs, the `check_clock_tree` command checks all scenarios that are enabled as clock tree synthesis scenarios for the following issues:

- Conflicting per-clock exception settings for each scenario
- Conflicting balancing settings in merged scenarios
- Conflicting multicorner-multimode interclock delay balancing settings

To generate a file containing suggested solutions for resolving many of the issues identified by the `check_clock_tree` command, use the `-solution_template` option, as shown in the following example:

```
icc_shell> check_clock_tree -solution_template cts_solution.tcl
```

For a complete list of issues for which the tool generates solutions, see the man page for the `check_clock_tree` command.

The tool generates multiple solutions for some issues, consisting of a default solution and an alternate solution. To limit the solution template file to contain only the solutions for specific types of issues and to select either the default or alternate solution for each type of issue, use the `-solution_for` option with the `-solution_template` option.

Use the {CTS-XXX-Y} format for the `-solution_template` option, where CTS-XXX represents the warning message ID number for the issue and Y represents the selection between the default and alternate solutions. For Y, use 1 to select the default solution or 2 to select the alternate solution.

The following example shows how to generate a solution template file that contains the default solutions for all the CTS-823 warnings and the alternate solutions for all the CTS-835 warnings:

```
icc_shell> check_clock_tree -solution_template cts_solution_1.tcl \
    -solution_for {CTS-823-1 CTS-835-2}
```

Resolving some issues can expose other issues in the design. After you have applied the solutions to your design, rerun the `check_clock_tree` command to check if new issues have been exposed.

Before you implement the clock trees, manually fix all issues for which the tool does not generate solutions. Each message generated by the `check_clock_tree` command has a detailed man page that describes how to fix the identified issue. You can improve the clock tree and timing QoR by fixing all the issues identified by the `check_clock_tree` command.

To see the current clock tree definition, generate the clock tree settings and exceptions reports, as described in “[Generating Clock Tree Reports](#)” on page 5-135.

Implementing 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, the tool 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, the tool 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.

[Table 5-14](#) describes the options you can specify for the `clock_opt` command.

Table 5-14 Clock Tree Synthesis and Optimization Options

Command option	GUI object	Description
N/A	“CTS, clock routing, extraction, optimization and fix hold time violations” option	Selects the default <code>clock_opt</code> behavior.
N/A	“Allow cell insertion and deletion” option	Selects the default <code>clock_opt</code> clock tree optimization behavior.
<code>-clock_trees</code>	“Clock Tree Names” box	Specifies a list of clocks to synthesize.
<code>-congestion</code>	“Congestion minimization” check box	Reduces congestion.
<code>-area_recovery</code>	“Recover area” check box	Enables area recovery during placement and timing optimization.
<code>-fix_hold_all_clocks</code>	“Fix hold time violations for all clocks” check box	Fix hold time violations.
<code>-in_place_size_only</code>	“Minimal ECO placement changes” option	Restrict placement and timing optimization to gate sizing only when there is enough space. You cannot use this option with <code>-size_only</code> . For more information, see “Performing Incremental Physical Optimization” on page 4-84 .
<code>-inter_clock_balance</code>	“Balance interclock delay” check box	Enables interclock delay balancing.
<code>-no_clock_route</code>	“Route clock nets” check box (uncheck for <code>-no_clock_route</code>)	Do not perform clock routing.
<code>-only_cts</code>	“CTS only (CTS, CTO, clock routing)” option	Do not perform extraction, optimization, or hold time fixing.
<code>-only_hold_time</code>	“Fix hold timing only after CTS” check box	Performs only hold time fixing. (Do not perform clock tree synthesis, clock tree optimization, clock routing, extraction, or optimization.)

Table 5-14 Clock Tree Synthesis and Optimization Options (Continued)

Command option	GUI object	Description
<code>-only_psyn</code>	“Incremental optimization only (clock routing, extraction, optimization and fix hold time)” option	Do not perform initial gate upsizing, clock tree synthesis, or clock tree optimization.
<code>-concurrent_clock_and_data</code>	N/A	Performs clock tree synthesis and optimization based on clock and data analysis.
<code>-incremental_concurrent_clock_and_data</code>	N/A	Performs incremental clock tree synthesis and optimization based on clock and data analysis.
<code>-operating_condition</code>	“Operating condition” list box	Specifies the operating condition to use for clock tree synthesis and optimization: <code>min</code> , <code>max</code> , or <code>min_max</code> .
<code>-optimize_dft</code>	“Reorder scan cells” check box	Enables scan chain optimization. For more information about scan designs, see “ Performing Placement and Optimization on Designs With Scan Chains ” on page 4-32.”
<code>-continue_on_missing_scandef</code>	“Continue with missing scan definitions” check box	Continues placement when the design contains scan chains but no SCANDEF data. By default, missing SCANDEF data causes the command to exit with an error message. Setting this option enables the placer to continue with a warning and results in reduced QoR.
<code>-power</code>	“Power optimization” check box	Performs power optimization. For multicorner-multimode designs, you must first use the <code>set_scenario_options</code> command to set the leakage scenarios before enabling power optimization.

Table 5-14 Clock Tree Synthesis and Optimization Options (Continued)

Command option	GUI object	Description
<code>-size_only</code>	“Sizing changes only” option	Restrict placement and timing optimization to gate sizing only. You cannot use this option with <code>-in_place_size_only</code> . For more information, see “ Performing Incremental Physical Optimization ” on page 4-84.
<code>-update_clock_latency</code>	“Update real and virtual clock latencies” check box	Enables updating of clock latency values.
<code>-insert_self_gating</code>	“Insert XOR self-gating” check box	Enables XOR self-gating. For more information, see “ Using XOR Self-Gating to Save Dynamic Power ” on page 5-30.

The `clock_opt` command does the following:

1. Synthesizes the clock trees.

Before implementing the clock trees, the tool 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, the tool might move the existing gates, including integrated clock-gating (ICG) cells, when this could improve QoR. To prevent the tool from moving existing gates, including integrated clock-gating cells, before clustering, set the `cts_move_clock_gate` variable to `false`.

The tool builds clock trees that meet the clock tree design rule constraints, while balancing the loads and minimizing the clock skew. In addition, the tool 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 block abstraction models.

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 affects the timing QoR.

To synthesize specific clocks, use the `-clock_trees` option with the `clock_opt` command.

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 5-98](#).

2. Optimizes the clock trees.

During clock tree optimization, the tool 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, the tool 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 5-101](#).

3. (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 5-39](#), and use the `-inter_clock_balance` option with the `clock_opt` command.

Note:

The tool performs interclock delay balancing by performing delay insertion at the clock root. If the clock root net has a don't buffer net exception, the tool 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 5-103](#).

4. Performs detail routing of the clock nets.

You can also perform detail routing of the clock nets as a standalone process by 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.

5. Performs RC extraction of the clock nets and computes accurate clock arrival times.
6. (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. The tool 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 5-103](#).

7. (Optional) Performs power optimization.

To perform physical optimization of the integrated clock-gating cells, power-aware placement, and leakage-power optimization during the `clock_opt` process, use the `-power` option with the `clock_opt` command. For more information about how to enable and perform power optimization during clock tree synthesis, see “[Optimizing for Power](#)” on [page 5-35](#).

8. (Optional) Reduces congestion.

By default, the `clock_opt` command does not perform congestion reduction. To reduce congestion and improve routability, use the `-congestion` option of the `clock_opt` command.

When you enable congestion reduction, by default, the tool considers scan net connections in the design. To ignore scan net connections during congestion reduction, use the `-ignore_scan` option. If your design contains scan chains but no SCANDEF data, the `clock_opt -congestion` command fails. To prevent the `clock_opt -congestion` command from failing when your design contains scan chains but no SCANDEF data, use the `-continue_on_missing_scandef` option. See “[Annotating the Scan Chain Information](#)” on [page 3-21](#). for more information about applying SCANDEF data.

9. (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.

You can also reorder the scan chains as a standalone process by using the `optimize_dft -clock_buffer` command.

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

11. Performs placement and timing optimization.

If you specify `-update_clock_latency`, the tool uses the adjusted input and output delays during placement and timing optimization. The tool 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-power optimization by using the `-power` option. For more information about these options, see [Table 5-14 on page 5-85](#).

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.

You can also run only placement and timing optimization as a standalone process, by using the `psynopt` command. For more information about the `psynopt` command, see [“Performing Incremental Physical Optimization” on page 4-84](#).

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.

To reduce the runtime and buffer count, you can set the `-effort` option to `high` with the `set_fix_hold_options` command to guide the hold fixing performed by the `clock_opt` command. By default, the `-effort` option is set to `medium`. Fixing hold time violations during preroute optimization also improves the QoR for postroute optimization.

Note:

The setting of the `-effort` option with the `set_fix_hold_options` command applies only to the preroute optimization stage.

13.(Optional) Performs clock tree synthesis and optimization based on clock and data analysis to improve timing.

To enable clock tree synthesis and optimization based on clock and data analysis use the `-concurrent_clock_and_data` option with the `clock_opt` command.

14.(Optional) Performs incremental optimization based on clock and data analysis to improve timing.

To enable incremental optimization based on clock and data analysis use the `-incremental_concurrent_clock_and_data` option with the `clock_opt` command.

To enable clock tree synthesis batch mode when you run the `clock_opt` command, set the `cts_clock_opt_batch_mode` variable to `true`, as shown in the following example:

```
icc_shell> set_app_var cts_clock_opt_batch_mode true  
icc_shell> clock_opt
```

Using batch mode with the `clock_opt` command improves runtime.

After running the `clock_opt` command, analyze the results as described in [“Analyzing the Clock Tree Results” on page 5-134](#).

Performing Multicorner-Multimode Clock Tree Synthesis and Optimization

To perform clock tree synthesis and optimization on a multicorner-multimode design, use the following steps:

1. Enable specific scenarios for clock tree synthesis by using the `-cts_mode` option with the `set_scenario_options` command.

The following example enables the S1 and S2 scenarios for clock tree synthesis:

```
icc_shell> set_scenario_options -scenarios {S1 S2} -cts_mode true
```

The specified scenarios are made active during clock tree synthesis and optimization. However, they are returned to their previous state, active or inactive, after clock tree synthesis and optimization is completed.

In addition to clock tree synthesis, the specified scenarios are also enabled for setup and hold optimization. To disable the scenario for setup optimization, use the `-setup false` option with the `set_scenario_options` command. To disable the scenario for hold optimization, use the `-hold false` option. When you disable a scenario for setup optimization, some commands are not supported in that scenario, and the tool issues a CTS-1130 error message when you try to run these commands. For a list of unsupported commands, see the man page for the CTS-1130 error message.

2. Specify which corners of the given scenarios should be considered for clock tree optimization by using the `-cts_corner` option with the `set_scenario_options` command.

The following example specifies the minimum corner of the S1 scenario and the maximum corner of the S2 scenario for clock tree optimization:

```
icc_shell> set_scenario_options -scenarios S1 -cts_corner min  
icc_shell> set_scenario_options -scenarios S2 -cts_corner max
```

3. Perform clock tree synthesis and optimization by using the `clock_opt` command.

When you run the `clock_opt` command, the tool synthesizes and optimizes clock trees across all scenarios enabled for clock tree synthesis and optimization. To see all the scenarios for which clock tree synthesis and optimization is performed, use the `report_scenario_options` command.

Performing Priority-Driven Clock Tree Synthesis and Optimization

You can prioritize the order in which you implement the clock trees by synthesizing and optimizing them sequentially and freezing each implemented clock tree.

To perform priority-driven clock tree synthesis and optimization for a multicorner-multimode design, use the following steps:

1. Enable priority-driven clock tree synthesis and optimization by setting the `cts_enable_priority_driven` variable to `true`.
2. Set the current scenario to the one in which you want to synthesize the desired clock and enable it for clock tree synthesis by using the `set_scenario_options -cts_mode true` command.
3. Specify the scenarios and the corresponding corners which should be considered for clock tree optimization by using the `set_scenario_options -cts_corner` command.
4. Synthesize and optimize the desired clock by using the `clock_opt` command with the `-clock_tree` option.
5. Freeze the clocks trees you implemented by using the `mark_clock_tree` command. Specify the clocks you wish to freeze by using the `-seq_freeze` option. This ensures that the implemented clock trees are not modified when you run the `clock_opt -clock_trees` command again.

6. Disable the current scenario for clock tree synthesis by using the `set_scenario_options -cts_mode false` command.

If you plan to implement the next clock in the same scenario, it is not necessary to disable the scenario for clock tree synthesis.

7. Disable the specified clock tree synthesis corner setting by using the `set_scenario_options -cts_corner none` command.

If you plan to use the same clock tree synthesis corner setting for the next clock you implement, it is not necessary to disable it.

8. Repeat steps 2 through 7 until all the clock trees are implemented in the desired order.

When removing clock trees, use the `remove_clock_tree -honor_seq_freeze` command to preserve the clock trees that were frozen by using the `mark_clock_tree -seq_freeze` command.

[Example 5-1](#) shows a script for running the priority-driven flow for a multicorner-multimode design. The design has three scenarios (S1, S2, and S3), and four clocks (CK1, CK2, CK3, and CK4). The script first synthesizes and optimizes the CK1 and CK2 clocks in the S1 scenario, followed by the CK3 and CK4 clocks in the S2 scenario.

Example 5-1 Script for Performing priority-driven Clock Tree Synthesis and Optimization for a Multicorner-Multimode Design

```

set_app_var cts_enable_priority_driven true

current_scenario S1
set_scenario_options -scenarios {S1} -cts_mode true
set_scenario_options -scenarios {S1 S2 S3} -cts_corner min_max
clock_opt -only_cts -no_clock_route -clock_trees {CK1 CK2}
mark_clock_tree -clock_trees {CK1 CK2} -seq_freeze
report_clock_tree -clock_trees {CK1 CK2}
set_scenario_options -scenarios {S1} -cts_mode false
set_scenario_options -scenarios {S1 S2 S3} -cts_corner none

current_scenario S2
set_scenario_options -scenarios {S2} -cts_mode true
set_scenario_options -scenarios {S1 S2 S3} -cts_corner min_max
clock_opt -only_cts -no_clock_route -clock_trees {CK3 CK4}
mark_clock_tree -clock_trees {CK3 CK4} -seq_freeze
report_clock_tree -clock_trees {CK3 CK4}
set_scenario_options -scenarios {S2} -cts_mode false
set_scenario_options -scenarios {S1 S2 S3} -cts_corner none

```

Performing Concurrent Clock and Data Optimization

Useful skew techniques improve timing QoR of a design by adjusting the clock arrival times to take advantage of the positive slack in the design. However, applying the useful skew technique while performing datapath optimization, which is referred to as concurrent clock and data (CCD) optimization, results in a bigger improvement in the timing QoR.

Concurrent clock and data optimization in the IC Compiler tool consists of the following stages:

- Timing-based clock tree synthesis

During this stage, the tool builds the clock tree based on the timing QoR with the goal of improving the worst negative slack (WNS) and total negative slack (TNS) of the design. The tool also performs datapath optimization to ensure that there are timing paths with positive slack that can be used by the useful skew technique to improve other paths.

- Post-clock-tree-synthesis concurrent timing optimization

During this stage, the tool performs useful-skew-based optimization and datapath optimization concurrently. The useful-skew-based optimization analyzes and adjusts the existing clock tree to improve the worst negative slack (WNS) of all the active scenarios enabled for setup analysis.

- Incremental concurrent clock and data optimization

During this stage, the tool performs useful-skew-based optimization and datapath optimization concurrently, using incremental optimization techniques. The tool works on all path groups in all active scenarios enabled for setup analysis and tries to improve the worst negative slack (WNS) of each path group.

- Postroute concurrent clock and data optimization

During this stage, this tool performs useful-skew-based optimization and datapath optimization concurrently, using topology-based optimization techniques. The tool works on all path groups in all active scenarios enabled for setup analysis and tries to improve the worst negative slack (WNS) of each path group. For more information, see [“Performing Focal Optimization” in Chapter 6](#).

The following topics describe how to setup and run concurrent clock and data optimization:

- [Setting Up for Concurrent Clock and Data Optimization](#)
- [Running Concurrent Clock and Data Optimization](#)

Setting Up for Concurrent Clock and Data Optimization

Before you perform concurrent clock and data optimization,

- Specify the concurrent clock and data optimization strategy settings by using the `set.concurrent_clock_and_data_strategy` command. For example,
 - To specify that the clock latencies of the boundary registers should not be adjusted, use the `-adjust_boundary_registers false` option.
 - To specify a collection of ports to ignore during boundary register identification, use `-ignore_ports_for_boundary_identification` option, when you use the `-adjust_boundary_registers false` option setting.
 - To ignore all boundary paths, use the `-ignore_boundary_path_timing true` option.
 - To ignore a specific path group, use the `-ignore_path_groups` option with the name of the path group you want ignored.
 - To specify that the tool should focus on both the worst negative slack (WNS) and total negative slack (TNS) when applying useful skew techniques at the post-clock-tree-synthesis concurrent timing optimization stage, use the `-tns_effort medium` option. The default for this option is `low` which corresponds to the default behavior where the tool focuses only on the worst negative slack (WNS).

- To specify that the tool should use less effort during concurrent clock and data optimization, use the `-effort medium` option. Specifying an `-effort medium` option setting improves runtime at the expense of a slight degradation in the timing QoR, compared to the default setting of `high`. However, you can regain this QoR when you perform postroute optimization later in the design flow.

To report all the options you have specified by using the `set.concurrent_clock_and_data_strategy` command, use the `report.concurrent_clock_and_data_strategy` command.

To reset all the options to their defaults, use the `reset.concurrent_clock_and_data_strategy` command.

- Apply all the post-clock-tree-synthesis optimization setting and constraints. For example,
 - Reduce the clock uncertainty.
 - Enable clock reconvergence pessimism removal.
 - Apply a critical range.
 - Enable hold fixing and specify the buffers to use for hold fixing.
- Specify that the tool should optimize clock gates when useful skew techniques are used during the timing-aware clock tree synthesis stage by setting the `skew_opt.optimize_clock_gates` variable to `true`.
- Ensure that clock tree synthesis sees all the critical timing paths in the design by enabling all timing critical scenarios for clock tree synthesis by using the `set.scenario_options -cts_mode true` command. Enable at least one scenario per timing mode. In addition, set the most timing critical scenario as the current scenario by using the `current_scenario` command.
- Specify that the tool considers float pin delays when updating the ideal clock latencies by setting the `update_clock_latency_consider_float_pin_delays` variable to `true`.

This variable affects the ideal clock latencies when you update them by using the `-update_clock_latency` option with the `clock_opt` command or by using the `update_clock_latency` command after you run the `clock_opt` command.

During the useful-skew-based optimization stage of concurrent clock and data optimization, the tool derives float pin delays to adjust the clock arrival times of sinks. When you update the clock latencies after concurrent clock and data optimization, if you do not consider the float pin delays derived by the tool, the updated ideal latencies can be unrealistic.

Running Concurrent Clock and Data Optimization

To run concurrent clock and data optimization, use the `clock_opt -concurrent_clock_and_data` command. Alternatively, you can run the `clock_opt -concurrent_clock_and_data -only_cts` command followed by the `clock_opt -concurrent_clock_and_data -only_psyn` command.

When you run concurrent clock and data optimization,

- Disable clock routing by using the `-no_clock_route` option. Concurrent clock and data optimization does not support detail routed clock trees. Route the clock trees after you complete concurrent clock and data optimization.
- Enable other optimization techniques relevant to your design by using options such as `-area_recovery`, `-power`, and `-congestion`.

After you run concurrent clock and data optimization, you can further optimize the design by running incremental concurrent clock and data optimization using the `clock_opt -incremental_concurrent_clock_and_data` command.

When you run incremental concurrent clock and data optimization,

- Only the `-area_recovery` and `-continue_on_missing_scandef` of the `clock_opt` command are supported.
- Clock routing is not performed. As a result, it is not necessary to explicitly disable clock routing by using the `-no_clock_route` option.

Note:

If you specify the `set.concurrent_clock_and_data_strategy -effort medium` setting and run the `clock_opt -concurrent_clock_and_data -only_psyn` command, the tool performs incremental concurrent clock and data optimization, in addition to the default optimization applicable for this command. As a result, it is not necessary to subsequently run the `clock_opt -incremental_concurrent_clock_and_data` command.

[Example 5-2](#) shows a script for running clock tree synthesis with concurrent clock and data optimization.

Example 5-2 Script for the Concurrent Clock and Data Flow

```
# Specify the concurrent clock and data strategy settings
set.concurrent_clock_and_data_strategy -adjust_boundary_registers false \
    -tns_effort medium

# Enable the optimization of clock gates
set_app_var skew_opt_optimize_clock_gates true

# Specify that the tool should consider float pin delays
# when updating the ideal clock latencies.
set_app_var update_clock_latency_consider_float_pin_delays true
```

```
# Set the current scenario to the most timing critical scenario
current_scenario $timing_critical_scenario

# Perform concurrent clock and data optimization
clock_opt -only_cts -concurrent_clock_and_data \
           -no_clock_route -congestion -update_clock_latency

# Perform further concurrent clock and data optimization
clock_opt -only_psyn -concurrent_clock_and_data \
           -no_clock_route -congestion -area_recovery -update_clock_latency

# Perform incremental concurrent clock and data optimization
clock_opt -incremental_concurrent_clock_and_data -area_recovery

# Route clock nets
route_zrt_group -all_clock_nets -reuse_existing_global_route true

# Update the clock latency
update_clock_latency

# Check the QoR
report_qor
```

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, the IC Compiler tool 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
- Clock Tree Routing

The script in [Example 5-3](#) provides an example of performing clock tree synthesis and optimization by using the standalone capabilities. The following sections provide details about these capabilities.

Example 5-3 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 5-35.

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 Clock Trees](#)” on page 5-84) and can also be run as a standalone process.

By default, IC Compiler clock tree synthesis 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, the IC Compiler tool

- 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 without balancing the skew if the `cts_fix_drc_beyond_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 startpoint to an endpoint with minimum delay to prevent DRC violations.

If your design has logical hierarchy, the tool 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.

The tool 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” box).

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.

Using Batch Mode to Reduce Runtime

When you run multiple `compile_clock_tree` commands on your design, the overhead of repeated clock tree synthesis preprocessing and postprocessing increases runtime. To reduce the `compile_clock_tree` runtime, you can enable the clock tree synthesis batch mode by using the `set_cts_batch_mode` command.

The following script shows an example of how to use batch mode with the `compile_clock_tree` command:

```
icc_shell> set_cts_batch_mode
icc_shell> compile_clock_tree -clock_trees c1
icc_shell> compile_clock_tree -clock_trees c2
icc_shell> report_cts_batch_mode
...
icc_shell> compile_clock_tree -clock_trees c3
...
icc_shell> reset_cts_batch_mode
```

The `reset_cts_batch_mode` command disables batch mode and invokes the postprocessing cleanup of excess clock trees that were inserted. The `report_cts_batch_mode` command displays whether the flow is in batch mode or normal mode.

Using batch mode for the `compile_clock_tree` command saves an average of 15 percent in runtime with a minimum impact on QoR. For best results, avoid using trigger commands that call clock tree synthesis preprocessing or postprocessing between the

`set_cts_batch_mode` and `reset_cts_batch_mode` commands. Some examples of trigger commands are the `skew_opt`, `get_attribute`, `set_attribute`, `insert_buffer`, `remove_buffer`, `save_mw_cel`, `set_clock_tree_references`, and `latency` commands.

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” box).

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 5-43.

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 5-50.

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.

Performing Clock Tree Optimization

Clock tree optimization improves the clock skew and clock insertion delay by applying additional optimization iterations. Clock tree optimization is performed during the `clock_opt` process (see “[Implementing Clock Trees](#)” on page 5-84) and can also be run as a standalone process before clock routing, after clock tree routing, or after detail routing. Typically, you would perform standalone clock tree optimization when timing optimization or incremental placement disturbs the clock skew or clock insertion delay.

To perform standalone clock tree optimization, use the `optimize_clock_tree` command or choose Clock > Optimize Clock Tree in the GUI. You can specify a list of clock trees, ports, or pins, but not hierarchical pins, as starting points of the clock network by using the `-clock_trees` option.

The IC Compiler tool provides the following incremental optimization capabilities:

- Buffer relocation by using the `-buffer_relocation` option.
- Buffer sizing by using the `-buffer_sizing` option.
- Delay insertion by using the `-delay_insertion` option.
- Gate relocation by using the `-gate_relocation` option.
- Gate sizing by using the `-gate_sizing` option.

Note:

During clock tree optimization, the tool 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.

By default, the `optimize_clock_tree` command assumes that the clock trees in your design are not routed. For unrouted clock trees, the `optimize_clock_tree` command can perform any of the incremental optimization capabilities. The default behavior is to perform all incremental optimizations.

If your clock trees are routed, you must explicitly specify the routing stage of the clock trees by using the `-routed_clock_stage` option of the `optimize_clock_tree` command.

Table 5-15 shows the supported routing stage keywords.

Table 5-15 Routing Stage Keywords

Value	Description
none	The clock trees are not routed.
global	The clock trees have global routing.
track	The clock trees have track assignment.
detail	The clock trees have detail routing.

For routed clock trees, `optimize_clock_tree` can perform only the sizing optimizations (default behavior is to perform both buffer sizing and gate sizing).

To run a subset of the available optimizations, you must explicitly specify the optimizations that you want. If you specify options that are not compatible with the routing status of your design, the tool generates an error message.

You only need to run postroute clock tree optimization one time. Running multiple postroute clock tree optimizations offers minimum or no advantage because you do not see skew improvement over the runtime.

For example, to perform only gate sizing on a routed design, enter the following command:

```
icc_shell> optimize_clock_tree -gate_sizing
```

After optimizing postroute clock trees, the `optimize_clock_tree` command performs engineering change order (ECO) routing and extraction. The type of ECO routing performed depends on the routing stage of the clock trees.

- For global routed clock trees, the tool performs incremental global routing.
- For track assigned clock trees, the tool performs detail routing (utilizing dangling wires).
- For detail routed clock trees, the tool performs detail routing (utilizing dangling wires) and performs two search-and-repair loops. To change the number of search-and-repair loops, use the `-search_repair_loop` option of the `optimize_clock_tree` command.

To disable ECO routing during `optimize_clock_tree`, specify the `-no_clock_eco_route` option.

By default, clock tree optimization uses the integrated clock global router to estimate the wire delay and capacitance for better correlation with postroute timing.

Running Interclock Delay Balancing

Interclock delay balancing balances the skew between a group of clock trees, either as part of the `clock_opt` process (see “[Implementing Clock Trees](#)” on page 5-84) or as a standalone process.

For information about defining the interclock delay balancing requirements, see “[Balancing Multiple Clocks](#)” on page 5-39.

By default, interclock delay balancing uses the integrated clock global router to estimate the wire delay and capacitance for better correlation with postroute timing.

To run standalone interclock delay balancing, use the `balance_inter_clock_delay` command or choose *Clock > Balance Interclock Delay* in the GUI. The `balance_inter_clock_delay` command honors the Elmore or Arnoldi delay model that is set by the `set_delay_calculation_options` command.

Note:

The `-max_target_delay` option of the `balance_inter_clock_delay` command does not support interclock delay balancing for multicorner-multimode designs.

Adjusting the I/O Timing

After implementing the clock trees, the tool can update the input and output delays to reflect the actual clock arrival times. When you adjust the I/O timing, the tool calculates the median insertion delay for each clock tree and applies these values as the clock latency. The Milkyway database and SDC constraints are automatically updated, so you can easily export this data to PrimeTime for detailed timing analysis.

To adjust the I/O timing,

- Run the `update_clock_latency` command.
or
- Specify the `-update_clock_latency` option when you run the `clock_opt` command.

The tool adjusts the I/O timing to achieve the accuracy of clock latency and to prevent false timing violations on I/O paths after clock tree synthesis in the following ways:

- For synthesized generated clocks, the network latency of a clock object is updated, but the source latency of the clock object is updated when its master clock is synthesized.
- For synthesized clocks, network latency is computed by using the median value of the clock propagation delay, that is, the arrival time relative to the clock root at all boundary registers.
- For virtual clocks defined with the same `create_clock` command, network latency is calculated using the clock propagation delay of the boundary registers clocked by the individual virtual clocks.

To adjust the I/O timing for virtual clocks, you must define the relationships between the virtual clocks and the real clocks before you adjust the I/O timing as follows:

```
icc_shell> set_latency_adjustment_options \
           -to_clock my_virtual_clock -from_clock my_real_clock
icc_shell> update_clock_latency
```

When you save your design in Milkyway format, the relationships defined by the `set_latency_adjustment_options` command are stored in the Milkyway design library.

When adjusting the I/O timing based on virtual clocks, the `update_clock_latency` command defines the clock latency for both the real clock and its associated virtual clocks as the median insertion delay of the real clock.

You can report the virtual clock definitions by using the `report_latency_adjustment_options` command. You can remove the virtual clock definitions by using the `reset_latency_adjustment_options` command.

Performing Clock Routing

After you finish clock tree optimization, you can perform clock routing using either the default router, Zroute, or the classic router.

Both Zroute and the classic router support the integrated clock global router and balanced-mode routing. To achieve the best correlation results, the tool uses the integrated clock global router and saves clock global routing information.

When using Zroute (the default router), use the `route_zrt_group -all_clock_nets` command to perform clock routing. You must also use the `-reuse_existing_global_route` option so that Zroute detects the clock global routing information in the Milkyway database and performs incremental global routing.

```
icc_shell> route_zrt_group -all_clock_nets \
    -reuse_existing_global_route true
```

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.

For information about using Zroute to perform clock routing, see [Chapter 6, “Routing Using Zroute”](#).

To enable the classic router rather than Zroute, set the following command:

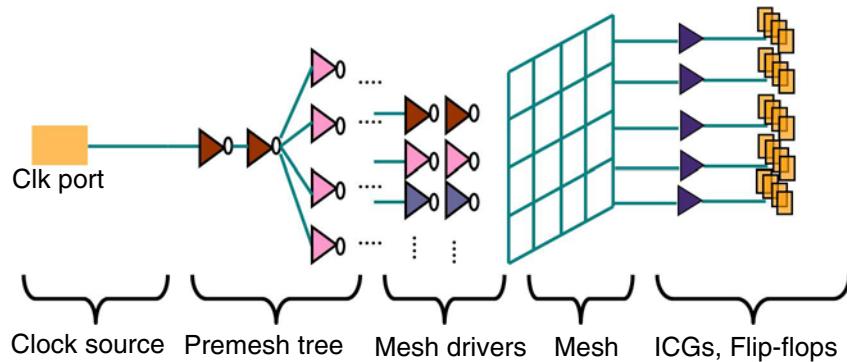
```
icc_shell> set_route_mode_options -zroute false
```

When using the classic router, you use the `route_group -all_clock_nets` command to perform clock routing. The `route_group` command can detect the clock global routing information in the Milkyway database and can set the global route incremental model automatically.

Implementing Clock Meshes

Clock meshes are homogeneous shorted grids of metal that are driven by many clock drivers. The purpose of a clock mesh is to reduce clock skew in both nominal designs and designs across variations such as on-chip variation, chip-to-chip variation, and local power fluctuations. A clock mesh reduces skew variation mainly by shorting the outputs of many clock drivers.

[Figure 5-21](#) shows the structure of a clock mesh. The network of drivers from the clock port to the mesh driver inputs is called the premesh tree. The network of shorted clock driver outputs is called the mesh.

Figure 5-21 Clock Mesh Structure

Using clock meshes provides the following benefits:

- Small skew variation, especially for high-performance designs
- Consistent design performance across variations
- Predictable results throughout both the design stage and ECO stage later
- Stability resulting from mesh grids being close to receivers

Using clock meshes has the following disadvantages:

- More routing resources are required to create clock meshes
- Higher power consumption during transitions on the parallel drivers driving the mesh

Prerequisites for Creating Clock Meshes

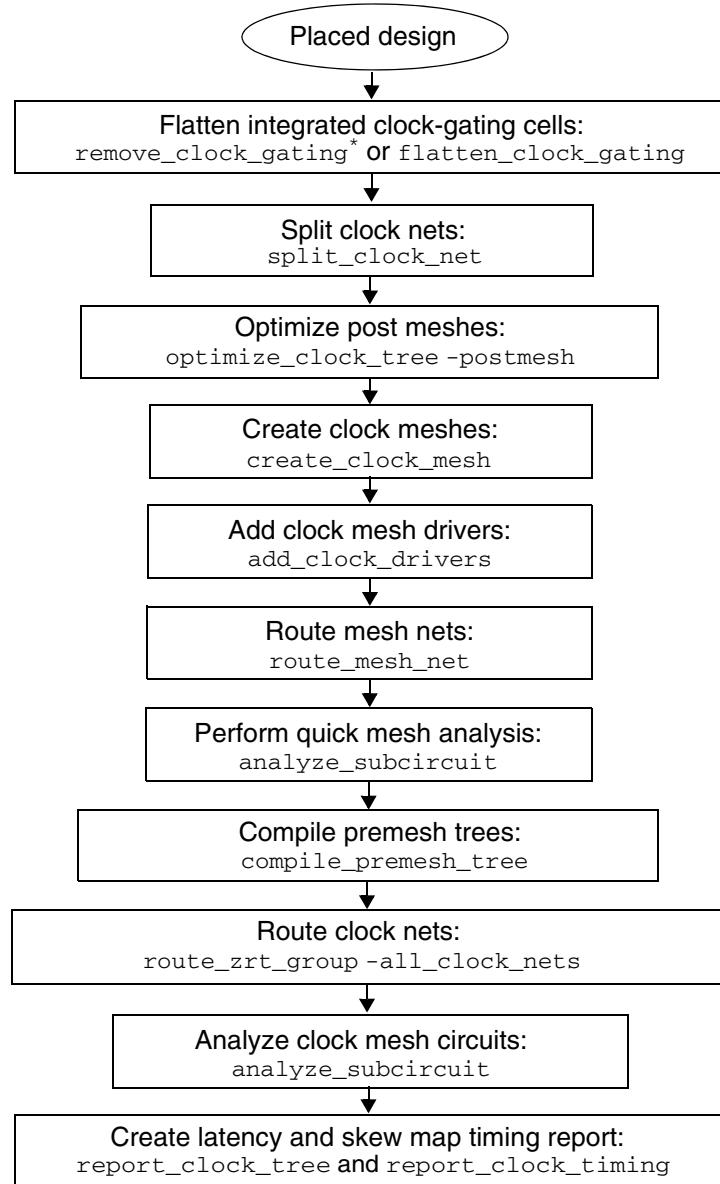
Before you run clock mesh commands, your design should meet the following requirements:

- The design should be mesh-conducive.
A basic mesh-conducive design contains at least one high-fanout clock net that has no more than two levels below the proposed mesh. If necessary, you can use the `flatten_clock_gating` command to flatten the circuitry under the proposed mesh.
- The design should have enough room to place mesh drivers near the mesh loads for driving the mesh optimally.
- To analyze clock mesh circuits, you must have the NanoSim or HSIM transistor models for all the clock mesh gates. A circuit simulator is needed because static timing tools cannot handle clock meshes.
- You should be able to run NanoSim from the shell where you invoke the IC Compiler.

The Clock Mesh Flow

Figure 5-22 shows the flow to use to create clock meshes.

Figure 5-22 IC Compiler Clock Mesh Flow



* This is a Power Compiler command.

The following sections describe the steps in the clock mesh flow:

- [Flattening the Logic of Integrated Clock-Gating Cells](#)
- [Splitting Clock Nets](#)
- [Creating Clock Meshes](#)
- [Adding Clock Mesh Drivers](#)
- [Routing Mesh Nets](#)
- [Performing Quick Mesh Analysis](#)
- [Compiling Premesh Trees](#)
- [Routing Clock Nets](#)
- [Analyzing Clock Mesh Circuits](#)

These sections describe how to implement flat clock meshes. To implement hierarchical clock meshes, see “[Implementing Hierarchical Clock Meshes](#)” on page 5-120.

Flattening the Logic of Integrated Clock-Gating Cells

Some designs have several levels of logic below the clock mesh, such as nested integrated clock-gating cells (ICGs). Integrated clock-gating cells are useful for reducing power consumption, but they can also introduce skew variation because of the number of levels.

To minimize skew induced by integrated clock-gating cells, you can

- Limit the depth of integrated clock-gating cells, preferably to a single level, by rerunning Power Compiler with appropriate options.
- Use the `remove_clock_gating` command in Power Compiler to remove integrated clock-gating cells.
- Use the `flatten_clock_gating` command to identify the integrated clock-gating cell that is driving a child integrated clock-gating cell. The command moves the child cell to a higher level of hierarchy in the circuitry where the primary mesh clock net is its clock input. To retain the same logic behavior, the command adds an AND gate and feeds both enable signals of the child cell and the parent cell to the AND gate.

Note:

Because the resulting design is more difficult to verify, use the `flatten_clock_gating` command only if you cannot rerun Power Compiler or use the `remove_clock_gating` command to remove integrated clock-gating cells.

Splitting Clock Nets

The clock mesh flow starts with a placed design where the clock network has not been built. As such, clock ports might be driving the clock inputs of registers, latches, macros, or float pins, resulting in a large number of clock sinks and causing large loads on the clock network. You can use the `split_clock_net` command to reduce the loads by clustering the clock sinks according to the placement of the loads, DRC requirements, and timing constraints.

If the clock network drives macros, use the `-isolate_float_pins` option with the `split_clock_net` command to isolate the float pin delays of the macro blocks from other pins while duplicating the clock gates. Next, you need to decide whether to connect the macro pins to the premesh tree or under the mesh. To determine the connection, you need to consider the pin capacitance and phase delay of the pins. You can use the `adjust_premesh_connection` command to make the correct connection.

The `split_clock_net` command automatically balances clock trees and performs clustering to improve skew. However, the command does not optimize the premesh trees because clock mesh designs impose strict timing constraints on the premesh trees. For high-performance clock mesh designs, use the following command to further reduce skew variation:

```
icc_shell> optimize_clock_tree -postmesh -mesh_net mesh_net \
    -clock_trees clock
```

where `mesh_net` is the high-fanout net associated with the clock to be implemented as a mesh.

The `optimize_clock_tree -postmesh` command performs optimization by sizing only.

Creating Clock Meshes

To create clock meshes, use the `create_clock_mesh` command and specify options such as the layers for the meshes, the numbers of straps, and so forth. Alternatively, you can choose Clock > Clock Mesh > Create Clock Mesh in the GUI. You should implement a simple mesh on an existing net that is connected to both a driving pin or port and some loads; otherwise, the net is removed during later stages.

The following command creates a 10 x 10 (horizontal straps x vertical straps) clock mesh that geometrically fits the tree structure of the existing clk clock net.

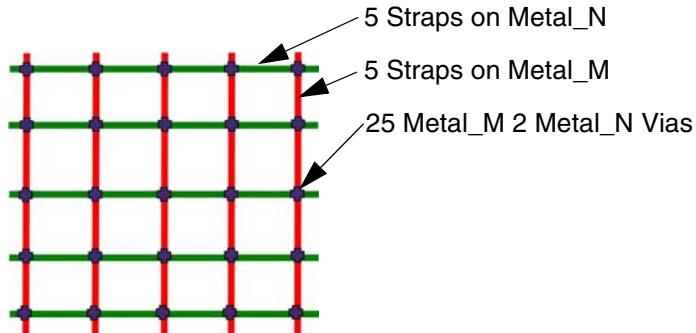
```
icc_shell> create_clock_mesh -load clk -net clk \
    -layers {METAL4 METAL5} -num_straps {10 10}
```

The `-load` option specifies the mesh wires for the load pins of the clk net and the `-layers` option specifies placing the mesh straps on METAL4 (a horizontal layer) and METAL5 (a vertical layer). You can use the `get_layers` command to find what layers are present in your design before using the `create_clock_mesh` command.

You can enable the `-check_only` option the first time you use the `create_clock_mesh` command to verify the command options and display the statistics of the clock mesh areas without actually creating the mesh.

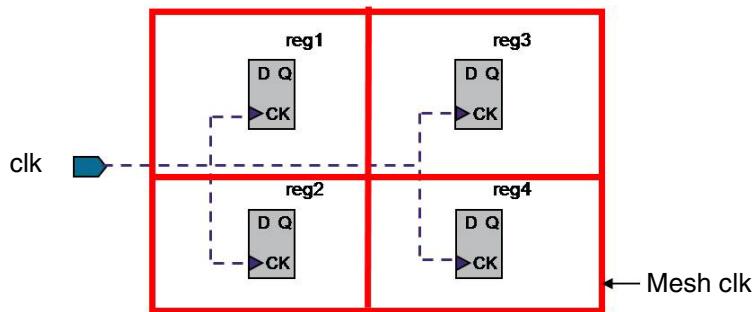
[Figure 5-23](#) shows an example of a resulting clock mesh using the `-num_straps {5 5}` option.

Figure 5-23 A Clock Mesh Created With the `-num_straps {5 5}` Option



[Figure 5-24](#) shows an example of a clock mesh that is created on the clk net.

Figure 5-24 A Clock Mesh Created on the clk Net



The process of creating clock meshes honors the fat wire rules that are specified in the technology file but does not honor the nondefault routing rules that are specified on the input nets. In addition, the command creates clock mesh wires with the clock strap attribute. To remove the clock mesh wires, enter

```
icc_shell> remove_clock_mesh -clockmesh clk_mesh
```

The `remove_clock_mesh` command allows you to remove different elements of the clock mesh, such as the premesh tree, the postmesh tree, and the entire clock mesh. [Table 5-16](#) describes the options of the command.

Table 5-16 remove_clock_mesh Command Options

To do this	Use this option
Specify a list of clock trees to be removed	<code>-clock_tree</code>
Remove the premesh tree	<code>-premesh</code>
Remove the routes of the premesh tree	<code>-route_only</code>
Remove the postmesh tree	<code>-postmesh</code>
Remove the routes of the clock mesh	<code>-clockmesh</code>

If you issue this command without any options, the tool removes the mesh and the routes of the mesh in the order of the clock mesh route, the premesh tree with all the routes, and the postmesh tree with all the routes.

Adding Clock Mesh Drivers

Clock mesh drivers directly drive the clock mesh loads that include the mesh straps created by the `create_clock_mesh` command, the loads directly driven by the mesh, and the comb routes used to connect the drivers and loads to the mesh. You can use the `add_clock_drivers` command either to add only the clock mesh drivers or to create a hierarchy of drivers that drive the mesh drivers, starting from the root of the clock tree. Alternatively, you can choose *Clock > Clock Mesh > Add Clock Drivers* in the GUI. The circuitry from the root to the mesh driver inputs is called the premesh tree. An ideal premesh tree propagates signals to the clock mesh with almost no skew, such as an H-tree, which has logic-level balancing of buffers and routes to minimize on-chip variation.

Before adding the drivers, you need to choose the name of the clock net, the type of drivers, a prefix for naming new cells and nets, and a new name for the clock mesh net if you want the resulting clock mesh on a different clock net.

For example, the following command adds one level of buffers directly on the clock mesh, and you can build the premesh tree later to drive the buffers by using the `compile_premesh_tree` command.

```
icc_shell> add_clock_drivers -load clk \
    -driver_type CQCLKBFX20 -prefix ccc \
    -configuration { \
        {-level 1 -boxes {10 10} -short_outputs -output_net_name CLK_MESH}}
```

The following example adds 100 drivers directly to the clock mesh at level 3, 25 drivers to level 2, and 4 drivers to level 1. The `-driver_type` option specifies the buffd7 buffer for all three levels, and the `-prefix` option adds a prefix of `ccc` to the new cells and nets.

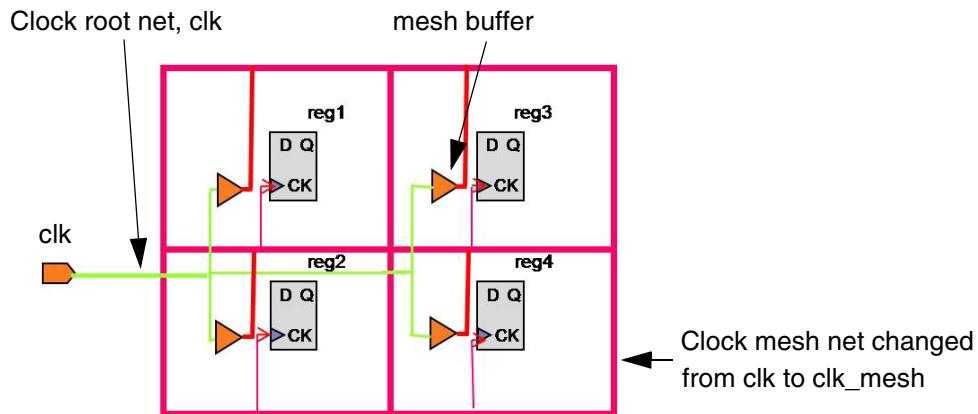
```
icc_shell> add_clock_drivers -load clk \
    -driver_type buffd7 -prefix ccc \
    -configuration { \
        {-level 1 -boxes {2 2}} \
        {-level 2 -boxes {5 5}} \
        {-level 3 -boxes {10 10} -short_outputs \
            -output_net_name clk_mesh -transfer_wires_from clk}}
```

Initially, the clock mesh was on the connected `clk` net. When the new mesh net named `clk_mesh` is created, the clock mesh is transferred to the `clk_mesh` net by using the `-transfer_wires_from` option.

Because level-3 buffers are driving the mesh directly, all their outputs are connected to create the new `clk_mesh` net.

[Figure 5-25](#) shows an example of a resulting mesh after inserting mesh drivers.

Figure 5-25 Clock Mesh With Mesh Drivers



Routing Mesh Nets

After you create the mesh drivers and premesh trees, use the router to connect the outputs of the mesh drivers to the nearest clock mesh and the mesh loads to the nearest mesh straps. By default, the tool uses Zroute as the default router. You can either use the `route_mesh_net` command or choose Clock > Clock Mesh > Route Mesh Net in the GUI to connect the mesh nets. For example,

```
icc_shell> route_mesh_net -net clk_mesh
```

By default, the `route_mesh_net` command uses comb topology when routing the clock mesh, which creates a thicker segment with many small segments coming off at

perpendicular direction. To create a fishbone structure, which includes backbones grown from the mesh stripes and many fingers connecting the drivers and loads to the backbones, use the `-mode fishbone` option when you run the `route_mesh_net` command.

To specify a layer to construct the backbone of a fishbone structure, use the `-backbone_layer` option. This option takes effect only when you specify a backbone direction by setting the `-backbone_dir` option to `horizontal` or `vertical` and choose the fishbone topology by setting the `-mode` option to `fishbone`. The default backbone layer can be any layer that is suitable for routing without DRC violations.

To honor the special routing rules that are specified on the mesh nets, use the following commands: `define_routing_rule` and `set_clock_tree_options`.

Performing Quick Mesh Analysis

You should analyze whether the clock mesh circuitry, which is partially built, meets your skew targets.

Prerequisites for Clock Mesh Analysis

You should analyze the partially built clock mesh circuitry to see if it meets the skew requirements.

- Create a temporary working directory to store the intermediate simulation files.
- Route the mesh nets by using the `route_mesh_net` command.
- Perform postroute RC extraction.

Before performing postroute RC extraction, ensure that the TLUPlus files are properly attached for extraction.

- Perform circuit simulation.

Before performing circuit simulations,

- Set up the transistor-level models for all gates of the clock network.
- Set up the search path for NanoSim or HSIM.

Quick Mesh Analysis

After you create the initial clock mesh, the clock root net is not yet built and is still a high-fanout net. You can use the following commands to perform a quick mesh analysis:

```
icc_shell> set_ideal_network [get_nets name_of_clk_root_net]
icc_shell> set from_pin [get_pins *ADD_CLK_DRIVER*/A]
icc_shell> analyze_subcircuit -from $from_pin -to CLK_MESH \
    -clock clkdec \
    -driver_subckt_files ${ORIG_DESIGN_DIR}/spice/cells.sp \
    -spice_header_files ${ORIG_DESIGN_DIR}/spice/header.txt
icc_shell> report_clock_tree
```

If error messages occur during the mesh analysis process, you must fix them before continuing analysis and annotation because they affect the accuracy of the analysis. The `analyze_subcircuit` command simulates the mesh circuits and back-annotates timing to the clock network. To see the clock skew, you can use the `report_clock_tree` command.

If you are not satisfied with the clock skew results, repeat the following steps until you get an acceptable result:

1. Remove the mesh structure and drivers by using the `remove_clock_mesh` command.
2. Re-create the mesh by using `create_clock_mesh`, using different mesh parameters.
3. Add mesh drivers by using the `add_clock_drivers` command.
4. Analyze the timing of the new mesh with the `analyze_subcircuit` command.
5. Report the clock skew by using the `report_clock_tree` command.

Compiling Premesh Trees

Creating premesh trees using the `add_clock_drivers` command can be challenging, especially for nonrectangular placement areas because you need to determine the number of logic levels and a placement area for the buffers. However, you can use the `add_clock_drivers` command to add only the mesh drivers and then use `compile_premesh_tree` command or choose Clock > Clock Mesh > Compile Premesh Tree in the GUI to build premesh trees.

For example, first add only one level of drivers to the clock mesh:

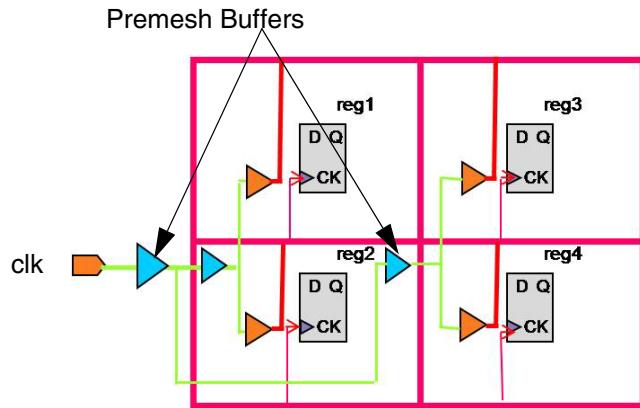
```
icc_shell> add_clock_drivers -load clk \
    -driver_type buffd7 -prefix ccc \
    -configuration { \
        {-level 1 -boxes {10 10} -short_outputs \
        -output_net_name clk_mesh -transfer_wires_from clk}}
```

Then, add the premesh tree drivers:

```
icc_shell> compile_premesh_tree -clock_tree clk
```

Figure 5-26 shows an example of a clock mesh with mesh drivers and premesh trees.

Figure 5-26 A Clock Mesh With Mesh Drivers and Premesh Trees



You can specify constraints and options for premesh tree synthesis by using the `set_clock_tree_options` command. By default, the `compile_premesh_tree` command enables the balancing of buffer levels for the premesh trees. To ensure a legal design with minimum skew, you can further improve the skew by running `optimize_clock_tree -premesh` after compiling the premesh trees.

Note:

The mesh root pin used for premesh clock tree synthesis must be the real clock root pin.

Alternatively, you can manually create a multilevel premesh tree by using `add_clock_drivers` (see “[Adding Clock Mesh Drivers](#)” on page 5-111 for an example) and then optimizing the premesh tree. For example, if the clock root of the premesh tree is `clk`, use the following command to optimize the premesh tree:

```
icc_shell> optimize_clock_tree -premesh \
-mesh_net [get_nets clk_mesh] -clock_trees clk
```

Creating H-, T-, or I-Shape Routes for Premesh Trees

The `add_clock_drivers` command can create a very regular premesh tree if you set the appropriate options, but the router cannot guarantee regular routes and low skew for the premesh tree. You get regular routes and low skew if you choose the H-, I-, or T-shape route by using the `route_htree` command or by choosing Clock > Clock Mesh > Route HTree in the GUI.

The `route_htree` command uses Zroute in comb mode to create H-, I-, or T-shape routes.

For a net that has one driver and four loads, you can use the `route_htree` command to create an H-shape route. For example, the following command uses Zroute to create an

H-shape route by using the M7 metal layer for vertical wires and the M6 layer for horizontal wires:

```
icc_shell> route_htree -nets [get_nets {ccc*L3_net* ccc*L2_net*}] \
-layers {M6 M7} -orientation {H}
```

Note:

To ensure DRC convergence, Zroute might choose layers other than M6 and M7.

For a net that has one driver and two loads, you can use the `route_htree` command to create an I-shape route. The following example shows how to create a rotated I-shape route:

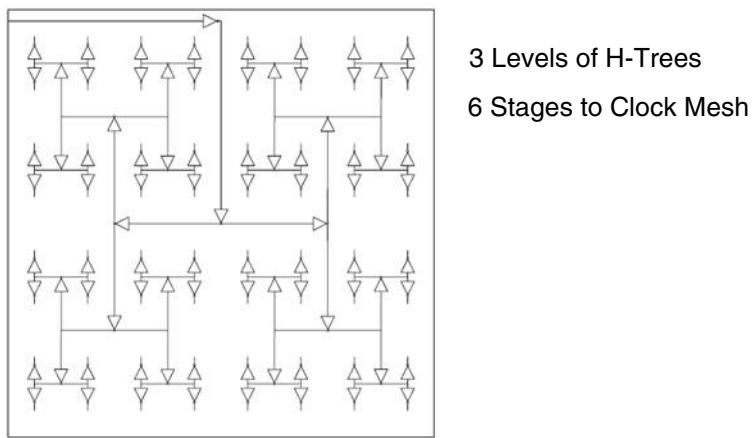
```
icc_shell> route_htree -nets [get_nets {ccc*L1_net*}] \
-layers {M6 M7} -orientation {I_90}
```

Both H- and I-shape routes produce low skew. Using a series of I-shape routes instead of H-shape routes requires more drivers but minimizes skew.

For a net that has one driver and three loads, you can use `route_htree` command to create a T-shape route.

[Figure 5-27](#) shows the premesh tree structure of H-shape routes.

Figure 5-27 H-Tree Structure



Use the `route_htree` command with the `-wire_balance` option to balance the total wire length of each net within the specified limits for each level of the tree. If a level has a combination of H-, T-, and I-shaped nets, the tool attempts to balance the length of each net to match the largest length.

You can use the `route_htree` command for wire-length-balanced routing in the following ways:

- To create wire-length-balanced-routing from the specified root to a specified clock mesh net, use the following syntax:

```
route_htree
  -root cell_name
  -wire_balance
  -mesh_net mesh_net_name
  ...
  . . .
```

The following example creates wire-length-balanced routing from the R_BUF_009_INST root to mesh_clock clock mesh by using the M6 layer for vertical wires and the M5 layer for horizontal wires:

```
icc_shell> route_htree -root R_BUF_009_INST -wire_balance \
  -mesh_net mesh_clock -layers {M5 M6}
```

- To create wire-length-balanced routing for a maximum of the specified levels from the specified root, use the following syntax:

```
route_htree
  -root cell_name
  -wire_balance
  -max_level num
  ...
  . . .
```

Note:

The `-max_level` option specifies the maximum number of levels to be processed when creating wire-length-balanced routing.

The following example creates wire-length-balanced routing for a maximum of three levels starting from the K_BUF_023_INST root by using the M6 layer for vertical wires and the M5 layer for horizontal wires:

```
icc_shell> route_htree -root K_BUF_023_INST -wire_balance \
  -max_level 3 -layers {M5 M6}
```

Routing Clock Nets

After you complete the routing of mesh drivers and premesh trees, perform detail routing of the clock nets by using the `route_zrt_group -all_clock_nets -reuse_existing_global_route true` command or choose Route > Net Group Routing in the GUI. See “[Routing Clock Nets](#)” on page [6-72](#) for more information.

Analyzing Clock Mesh Circuits

Clock mesh circuitry drives multiple loads across a wide area with many drivers, which are fundamentally different from the normal usage of CMOS logic gates. Additionally, clock mesh designs require higher timing accuracy than other designs to reduce skew variation. To analyze a clock mesh design, you use the RC extraction results followed by circuit simulation and then back-annotate the delay and transition time results to the IC Compiler static timing analyzer.

Before you analyze clock mesh circuits,

- Your design must meet the prerequisites for clock mesh analysis.
See “[Prerequisites for Clock Mesh Analysis](#)” on page 5-113 for more information.
- You must complete the routing of the clock network.
- If you want to use StarRC for extraction and for signoff accuracy, define StarRC in the search path and include the nxtgrd files.

Analyzing clock meshes consists of the following steps:

1. Traverse the loads that you specify from the clock root to the clock mesh net.
2. Perform RC extraction on the clock meshes.
3. Store the extraction results in the directory that you specify.
4. Generate the SPICE files to simulate the clock meshes:
 - Command files
 - Measurement files
 - Input pin capacitance and mesh receiver models
5. Invoke NanoSim or HSIM to perform simulation.
6. Annotate the timing analysis results from NanoSim to IC Compiler.

You can use the `report_clock_tree` or `report_clock_timing` command to see the timing results back-annotated from the `analyze_subcircuit` command.

To analyze clock mesh circuits, use the `analyze_subcircuit` command or choose Clock > Clock Mesh > Analyze Subcircuit in the GUI. For example,

```
icc_shell> analyze_subcircuit -name meshdir \
    -from clkroot -to clk_mesh -clock clk \
    -analysis_mode max -spice_header_files orig_data/header.txt \
    -driver_subckt_files orig_data/cells.sp
```

The command invokes extraction on the clock tree and places the results in a directory named meshdir that you specify by using the `-name` option. In addition, a CSV file records the latency and transition times for all input and output pins that are analyzed.

After the routing is complete, you can choose StarRC instead of `extract_rc` to perform extraction for signoff accuracy by specifying the `-starrcxt_map_file` and `-starrcxt_nxtgrd_file` options.

The `analyze_subcircuit` command supports several simulators: NanoSim (`nanosim`), HSIM (`hsim`), the 32-bit version of HSPICE (`hspice`), and the 64-bit version of HSPICE (`hspice64`). NanoSim is the default simulator; however, you can select the other simulators by using the `-simulator` option. The executable for the simulator you specify must be available in the UNIX search path.

The command provides three analysis modes: maximum (`max`), minimum (`min`), and maximum followed by minimum (`max_then_min`). The example uses the maximum mode for analysis. If you specify `-analysis_mode max_then_min`, the simulator performs analysis using the maximum corner first followed by the minimum corner.

You can also perform multicorner-multimode (MCMM) analysis on the clock mesh circuits by using the `-configuration` option. For example,

```
icc_shell> analyze_subcircuit -to clk \
    -driver_subckt_files max_spice_model \
    -spice_header_files header_file \
    -configuration { \
        {-scenario_name scenario1 \
            -max_driver_subckt_files max_file1 \
            -max_spice_header_files header_max1 \
            -min_driver_subckt_files min_file1 \
            -min_spice_header_files header_min1} \
        {-scenario_name scenario2 \
            -max_driver_subckt_files max_file2 \
            -max_spice_header_files header_max2 \
            -min_driver_subckt_files min_file2 \
            -min_spice_header_files header_min2}}
```

You must specify the scenario name that the `create_scenario` command uses. If you do not specify either `-max_driver_subckt_files` or `-max_spice_header_files` for the `-configuration` option, the command uses the files specified by the `-driver_subckt_files` or `-spice_header_files` option.

If your clock tree has multiple-input cells other than integrated clock-gating cells, such as multiplexers, you need to set the inputs explicitly by using the `-tie_high` and `-tie_low` options of the `analyze_subcircuit` command. The options provide control for the combinational gates in the premesh tree or other circuitry where the tool cannot identify an enable value automatically. For example,

```
icc_shell> analyze_subcircuit -tie_high [get_lib_pins my_lib/CT_CEL/E] \
    -name meshdir
```

Optimizing Meshes With Macros

If your mesh circuitry has macro pins or pins with nonzero phase delays that are connected to the clock, use the `adjust_premesh_connection` command immediately after the `split_clock_net` command to analyze and connect those pins to the premesh trees. Alternatively, you can choose Clock > Clock Mesh > Adjust Premesh Connection in the GUI.

As shown in [Figure 5-22 on page 5-107](#), the command sequence between the `split_clock_net` command and the second `analyze_subcircuit` command treats the output pin of the isolation buffer that is inserted by the `adjust_premesh_connection` command as the clock root. After the `analyze_subcircuit` step, connect the macro pins to the premesh tree and balance the input pin of the isolation buffer to ensure low skew by using the following command:

```
icc_shell> adjust_premesh_connection -root clock_root -premesh
```

During this process, the macro pins and the input pin of the isolation buffer are modeled as float pins. (Clock tree synthesis and analysis stops when encountering a float pin.) The input of the isolation buffer is annotated with the delay obtained from the circuit analysis, and then clock tree optimization is invoked to balance the timing of the float pins with the original clock root.

Implementing Hierarchical Clock Meshes

You can create a clock mesh for a design with physical hierarchy by using block abstractions for the lower-level blocks.

Before you begin creating a clock mesh for a hierarchical design, complete the following steps:

1. Complete the hierarchical design planning of the design, and create the Milkyway design libraries for the top-level and lower-level blocks.
2. Perform placement and optimization for the design.
3. Identify blocks that are clock mesh conducive. For more details on mesh conducive designs, see [“Prerequisites for Creating Clock Meshes” on page 5-106](#).

To create a clock mesh for the hierarchical design, perform the following steps:

1. Create clock meshes for the blocks that you identified as clock mesh conducive by using the clock mesh flow outlined in [Figure 5-22](#), and complete the routing for the blocks.
2. Use the `create_block_abstraction` command to generate the block abstraction for the block and save it using the `save_mw_cel` command.
3. Create the FRAM view for the block to be used at the top level.

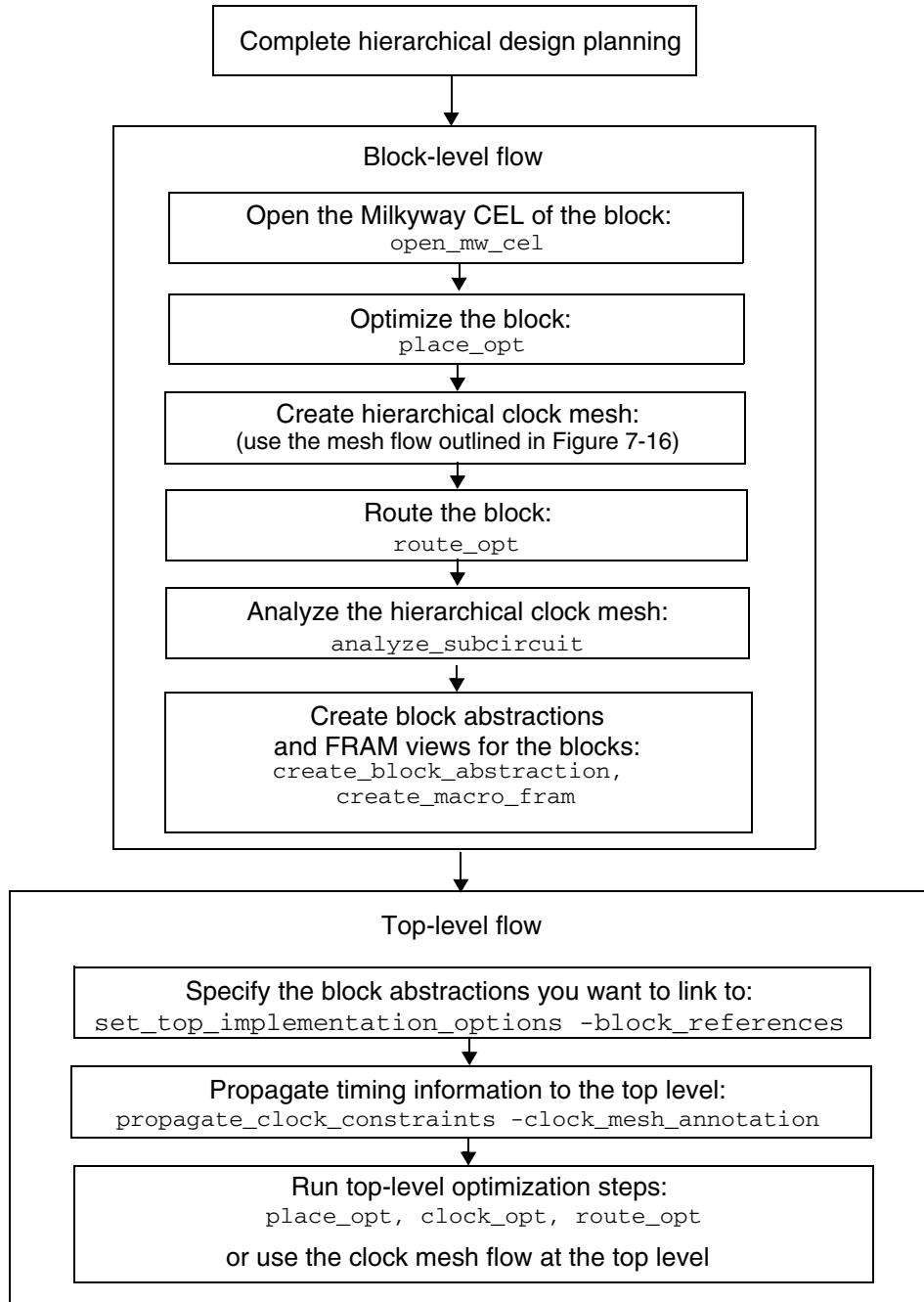
4. Return to the top level and specify the block abstraction to which you want link to by using the `set_top_implementation_options -block_references` command.
5. Propagate the block-level timing to the top level, and then proceed to the top-level flow. This is a required step in the hierarchical clock mesh flow.

```
icc_shell> propagate_clock_constraints -clock_mesh_annotation
```

If you change the mesh structure to refine the result after creating the block abstraction, you need to run the `analyze_subcircuit` command again and create a new block abstraction.

[Figure 5-28](#) shows the design flow for implementing hierarchical clock mesh.

Figure 5-28 Design Flow for Implementing Hierarchical Clock Mesh



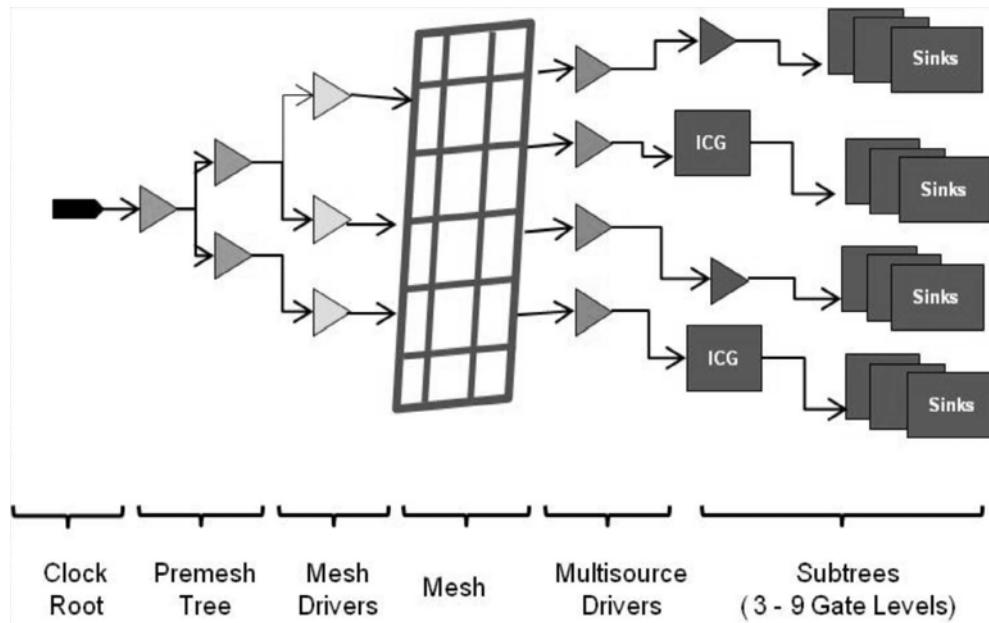
Implementing Multisource Clock Trees

A multisource clock tree combines the benefits of a conventional clock tree and a pure clock mesh. These benefits include

- Higher performance than a conventional clock tree
- Better tolerance to on-chip variation than a conventional clock tree
- Better multicorner performance than a conventional clock tree
- Less power consumption than a pure clock mesh
- More tolerant of complex clock structures, including deeper clock gating topologies, than a pure clock mesh

[Figure 5-29](#) shows the structure of a multisource clock tree. It consists of a mesh driven by a premesh tree, which is used to distribute the clock throughout the layout. Multisource drivers connect to the mesh at a limited number of locations. These locations are referred to as taps. The clock tree structure driven by the mesh consists of subtrees, each driven by a multisource driver (tap). Each subtree can have up to nine levels of logic. However, as the number of logic levels in the subtrees increases, the tolerance to on-chip variation decreases.

Figure 5-29 Multisource Clock Tree Structure



A multisource clock tree structure is suitable for large designs with sinks distributed throughout the layout.

Note:

For a multivoltage design, you can create the mesh only in a single voltage area with only one type of multisource drivers which is compatible with the voltage area.

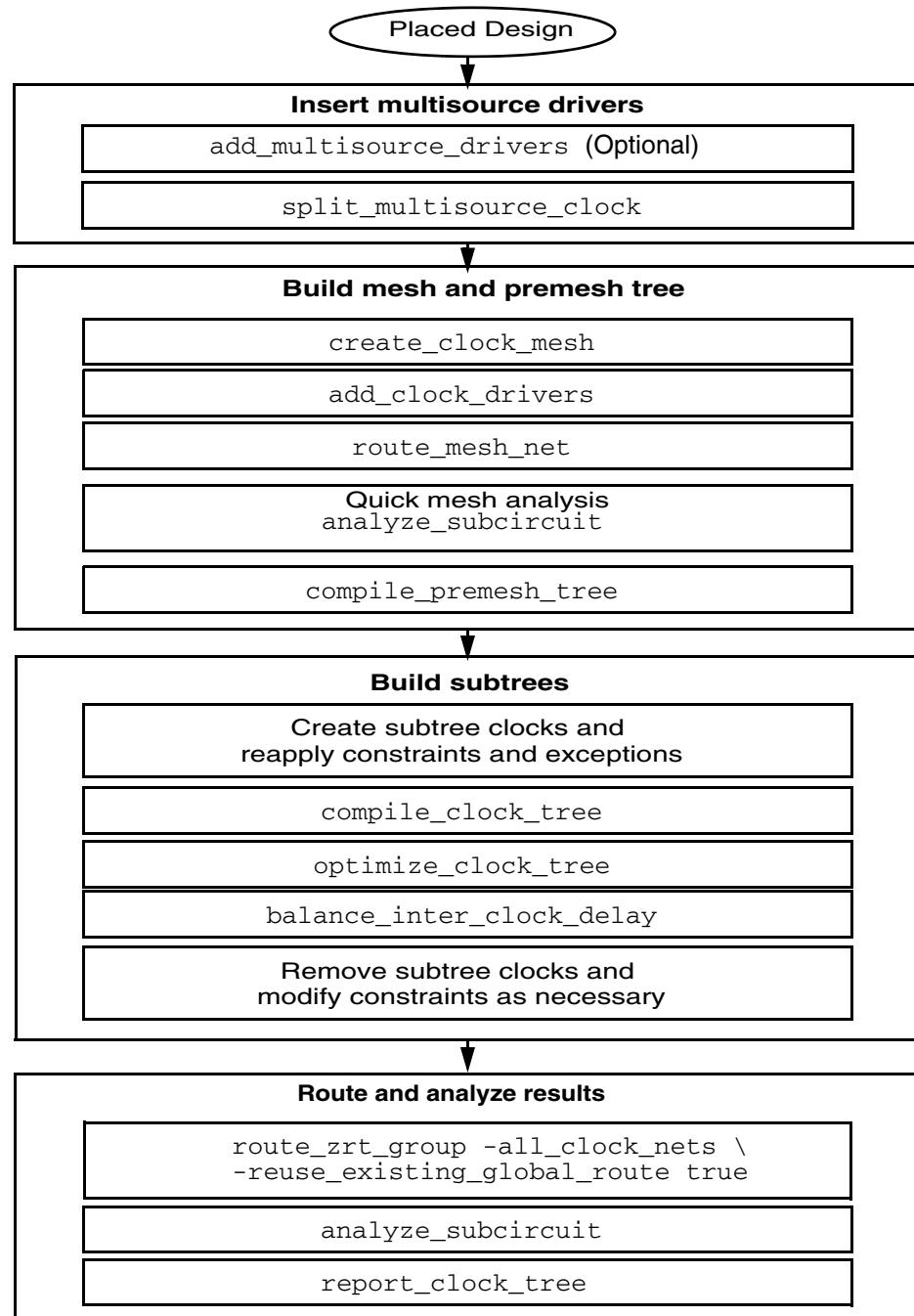
Multisource Clock Tree Synthesis Flow

[Figure 5-30](#) shows an overview of the multisource clock tree synthesis flow.

The following topics describe the tasks you need to perform to implement a multisource clock tree:

- [Inserting Multisource Drivers](#)
- [Building the Mesh and Premesh Tree](#)
- [Building the Subtrees](#)
- [Routing Clock Nets and Analyzing Results](#)

Figure 5-30 Overview of the Multisource Clock Tree Synthesis Flow



Inserting Multisource Drivers

Before you insert multisource drivers, you must perform the following tasks:

1. Create all scenarios and specify all timing constraints and settings.
2. Specify all clock tree synthesis and optimization setting, references, exceptions, and constraints.
3. If your design contains block abstractions, model the clock delays within the block abstractions by specifying float-pin exceptions at their clock inputs.

The `split_multisource_clock` command, which you use when you synthesize multisource drivers, does not recognize the clock network within block abstractions. However, after you run `split_multisource_clock` command, remove these float-pin exceptions.

4. Set the most timing-critical scenario as the current scenario.

The `split_multisource_clock` command works only on the current scenario.

5. Run the `update_timing` command to ensure the timing information is up-to-date.

To insert multisource drivers, use one of the following two methods:

- If you know the exact number and locations of the multisource drivers that your design requires, use the method described in “[Manually Inserting Multisource Drivers](#)” on [page 5-126](#).
- If you want the tool to insert the multisource drivers based on targets that you specify, use the method described in “[Automatically Synthesizing Multisource Drivers](#)” on [page 5-128](#).

Manually Inserting Multisource Drivers

If you know the number and locations of the multisource drivers that your design requires, to manually insert them and distribute the sinks among the drivers, use the following steps:

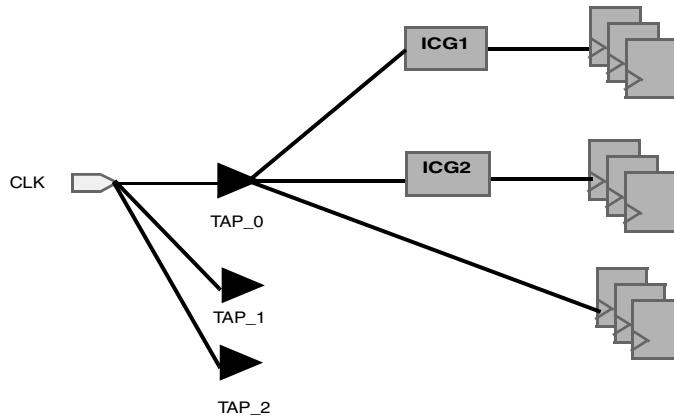
1. Add the multisource drivers to the clock net by using the `add_multisource_drivers` command.

If you know the exact locations, list them explicitly with the `-tap_locations` option. If you want the multisource drivers to follow a pattern of preexisting objects, such as vias, cells, port shapes, and net shapes, use the `-tap_markers` option, as shown in the following example:

```
icc_shell> add_multisource_drivers -mesh_net CLK \
    -prefix TAP_ -driver_type my_lib/clkbuff10 \
    -tap_markers [get_vias -of clk -filter {route_type=="Clk Strap"}]
```

When you run the `add_multisource_drivers` command, one multisource driver drives the entire clock network while the output of the other multisource drivers are left unconnected, as shown in [Figure 5-31](#).

Figure 5-31 Clock Network After Adding the Multisource Drivers



2. (Optional) Assign specific sinks to a specific multisource driver by using the `set_multisource_options` command, as shown in the following example:

```
icc_shell> set_multisource_options -force_tap TAP_2/z \
-sinks [get_pins U/my_reg*/CK]
```

To verify the sink assignments specified by using the `set_multisource_options` command, use the `report_multisource_options` command

3. Redistribute the sinks among all the multisource drivers by using the `split_multisource_clock` command

When you use the `split_multisource_clock` command during the manual insertion flow, you must specify

- The output pin of the multisource driver that is driving all the sinks by using the `-from` option.
- The output pins of all multisource drivers, including the one that is driving all the sinks, by using the `-driver_pins` option.

The following example shows how to specify the `-from` and `-driver_pins` option for the clock network shown in [Figure 5-31](#):

```
icc_shell> split_multisource_clock -from TAP_0/z \
-driver_pins [get_pins TAP_*/z]
```

When you use the `split_multisource_clock` command during the manual insertion flow, you can

- Minimize the effects of on-chip variation by using the `-ocv_aware` option.
The `-ocv_aware` option has no effect on sinks you assign to specific drivers by using the `set_multisource_options` command.
For more information, see “[Minimizing the Effects of On-Chip Variation](#)” on [page 5-130](#).
- Copy the constraints of any preexisting gates that are cloned by using the `-copy_constraints` option.
For more information about how to handle constraints related to the replicated gates, see “[Copying Constraints for Cloned Gates](#)” on [page 5-131](#).
- Remove all unused multisource drivers by using the `-remove_unused_taps` option.

Automatically Synthesizing Multisource Drivers

To automatically synthesize multisource drivers and distribute the sinks among them, use the following steps:

1. Manually insert a prototype driver on the clock-mesh net by using the `insert_buffer` command, as shown in the following example:

```
icc_shell> insert_buffer [get_nets CLK] my_lib/clkbuff10 \
    -new_cell_names TAP
```

You do not have to specify a location for the prototype driver. If there are sinks that you want to be driven directly by the clock-mesh net, isolate them from the prototype driver.

The tool clones the prototype driver and redistributes the sinks among the cloned drivers during the next step of the flow. Because the tool does not resize the drivers during this process, pick a prototype driver with the appropriate drive strength.

2. Clone the prototype driver and redistribute the sinks by using the `split_multisource_clock` command with the `-tap_synthesis` option.

You must specify one of the following two settings with the `-tap_synthesis` option:

- `split`

When you specify this setting, if the tool clones any integrated clock-gating cells, it can assign one or more clones of a given clock-gating cell to the same multisource driver.

- unique_icgs

When you specify this setting, if the tool clones any integrated clock-gating cells, it can assign no more than one clone of a given integrated clock-gating cell to the same multisource driver.

When you use the `-tap_synthesis` option, you must specify

- The input pin of the prototype driver by using the `-tap_input` option.
- The root of the clock being processed by using the `-root` option.

The following example shows how to specify the `-tap_input` and `-root` options:

```
icc_shell> split_multisource_clock -tap_synthesis split \
    -tap_input TAP/A -root CLK
```

When you use the `-tap_synthesis` option, you can

- Specify one of the following targets for the tool to use as a guideline:
 - The maximum distance from the driver to the sinks by using the `-target_max_distance` option.
 - The maximum number of levels by using the `-target_max_levels` option.
 - The maximum latency by using the `-target_max_latency` option.

When you synthesize the multisource drivers, the tool does not attempt to balance the number of sinks or latency under each multisource driver nor among the different drivers. It only tries to control the maximum distance, latency, or number of levels for each driver, using the target you specify as a guideline. However, there is no guarantee that the tool will meet this target when you build the subtrees for the sinks driven by each driver, in the next step of the multisource clock tree synthesis flow.

- Minimize the effects of on-chip variation by using the `-ocv_aware` option.

You can use the `-ocv_aware` option only with the `-tap_synthesis unique_icgs` setting. You cannot use the `-ocv_aware` option with the `-tap_synthesis split` setting. For more information, see “[Minimizing the Effects of On-Chip Variation](#)” on page 5-130.

- Copy the constraints of any preexisting gates that are cloned by using the `-copy_constraints` option.
For more information about how to handle constraints related to the replicated gates, see “[Copying Constraints for Cloned Gates](#)” on page 5-131.
- Display the locations of the drivers in the GUI without actually inserting them by using the `-check_only` option.

To remove the displayed locations from the GUI, use the `-remove_display` option.

To connect some sinks to a specific multisource drivers, but automatically synthesize the rest of the drivers and distribute the remaining sinks among them, use the following steps:

1. Add the multisource drivers, including the prototype driver, by using the `add_multisource_drivers` command and specify their locations by using the `-tap_locations` option.
For example, to assign specific sinks to one driver, you must add two drivers, one of which is used as the prototype driver for the rest of the sinks.
2. Assign specific sinks to a specific multisource driver by using the `set_multisource_options` command.
3. Synthesize the multisource drivers and distribute the sinks appropriately by using the `split_multisource_clock` command with the `-tap_synthesis` option.

When you run this command, the tool clones all drivers, including the prototype driver and the drivers to which you have assigned specific sinks, as necessary. However, the sinks you assigned to a specific driver are only distributed among that driver and its clones, even when you use the `-ocv_aware` option. The remaining sinks are distributed among the prototype driver and its clones.

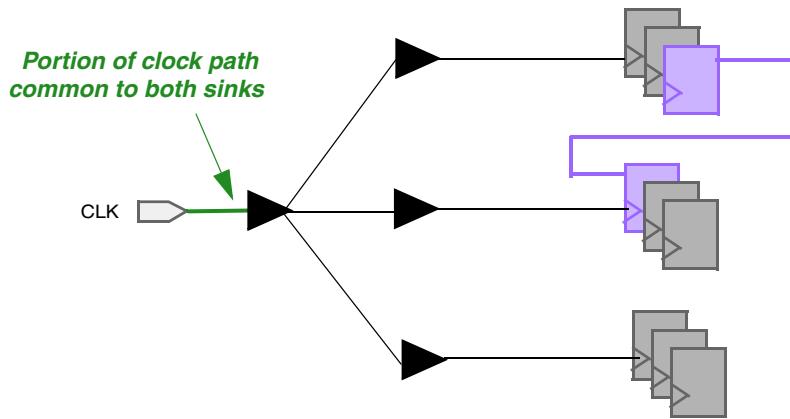
The following example shows how to add two drivers named TAP_0 and TAP_1, assign specific sinks to the driver named TAP_1, and perform automatic synthesis for the remaining sinks that are driven by the driver named TAP_0:

```
icc_shell> add_multisource_drivers -mesh_net CLK \
    -prefix TAP_ -driver_type my_lib/clkbuff10 \
    -tap_locations {{400 500} {600 700}}
icc_shell> set_multisource_options -force_tap TAP_1/Z \
    -sinks [get_pins IMG_BLOCK/*/CK]
icc_shell> split_multisource_clock -tap_synthesis split \
    -tap_input TAP_0/A -root CLK
```

Minimizing the Effects of On-Chip Variation

When assigning sinks to multisource drivers, by default, the tool does not consider the timing relationships between the registers. Two registers that have a timing path between them might be assigned to two different drivers. As a result, the timing path between the registers can be affected by on-chip variation because only a very small portion of the clock path is common to both registers, as shown in [Figure 5-32](#).

Figure 5-32 Impact on Timing Paths After Distributing the Sinks



To minimize the effects of on-chip variation by considering the timing between registers when assigning sinks to the drivers, use the `-ocv_aware` option with the `split_multisource_clock` command.

When you use the `-ocv_aware` option, the tool

- Only considers the setup timing in the current scenario.
- Does not consider any timing derate settings.
- Does not consider the timing constraints and paths related to the integrated clock-gating cells.

Copying Constraints for Cloned Gates

When splitting the clock network, the tool clones the preexisting gates if necessary. By default, the tool does not copy the clock tree exceptions, constraints, and attributes from the original gates to the clone. To copy the constraints and settings, use the `-copy_constraints all` option with the `split_multisource_clock` command.

The `split_multisource_clock` command works on the current scenario. However, when you specify the `-copy_constraints all` option, the tool copies the constraints and settings of all the scenarios, both active and inactive. Therefore, ensure that you create all the scenarios for which the constraints must be copied before you run the `split_multisource_clock` command.

The `split_multisource_clock -copy_constraints all` command does not copy the following constraints and settings:

- UPF constraints
- Attributes defined by using the `define_user_attribute` command

After you run the `split_multisource_clock` command, manually replicate the constraints and settings that are not handled automatically by the tool.

Limitations of the `split_multisource_clock` command

The `split_multisource_clock` command has the following limitations:

- For multicorner-multimode designs, it works only on the current scenario.
- It does not consider UPF constraints.
- It does not support clocks trees that are reconvergent.
- It does not support discrete clock gates.

Building the Mesh and Premesh Tree

Build the mesh and premesh tree by carrying out the following tasks:

1. Create the clock mesh by using the `create_clock_mesh` command. See “[Creating Clock Meshes](#)” on page 5-109 for more information.
2. Add the clock mesh drivers by using the `add_clock_drivers` command. See “[Adding Clock Mesh Drivers](#)” on page 5-111 for more information.
3. Route the mesh nets by using the `route_mesh_net` command. See “[Routing Mesh Nets](#)” on page 5-112 for more information.
4. Analyze the partially built clock mesh circuitry to see if it meets your skew targets by using the `analyze_subcircuit` command. See “[Performing Quick Mesh Analysis](#)” on page 5-113 for more information.
5. Compile the premesh tree by using the `compile_premesh_tree` command. See “[Compiling Premesh Trees](#)” on page 5-114 for more information. For uniform clock distribution at the top level you can build an H-tree as the premesh structure, instead of using the `compile_premesh_tree` command. See “[Creating H-, T-, or I-Shape Routes for Premesh Trees](#)” on page 5-115 for more information.

Building the Subtrees

Create a clock at the output of each of the multisource drivers by using the `create_clock` command. The characteristics of the newly created clocks, such as period and waveform, should be the same as that of the primary clock for which the multisource clock tree is being synthesized.

```
icc_shell> create_clock -period 0.25 \
           -name tap_0 [get_pins tap_0/z]
```

If there are any generated clocks in the subtrees redefine them by using the `create_generated_clock` command. Use the corresponding subtree clock as the master clock when you redefine a generated clock. Reapply all clock constraints, clock tree synthesis settings, and clock tree optimization settings, with respect to the subtree clocks. For a design with multiple scenarios, carry out these tasks for each scenario.

Compile and optimize all the subtrees driven by the multisource drivers. Then optimize across all the subtrees.

```
icc_shell> compile_clock_tree -clock_trees [get_clocks tap*]
icc_shell> optimize_clock_tree -clock_trees [get_clocks tap*]
icc_shell> balance_inter_clock_delay -clock_trees [get_clocks tap*]
```

For multimode clock tree synthesis, the clock names must be specified using the convention `scenario_name@clock_name` when synthesizing, optimizing, and balancing the interclock delays. The order in which you specify the clocks for synthesis and optimization determines the final clock tree QoR.

After you have built all the subtrees, remove all the subtree clocks that you created at the output of each multisource driver by using the `remove_clock` command. Redefine any generated clock that you defined using the subtree clocks as the master clock.

Routing Clock Nets and Analyzing Results

Route the clock nets and analyze the results by carrying out the following tasks:

1. Route the clock nets by using the `route_zrt_group -all_clock_nets -reuse_existing_global_route true` command or choose Route > Net Group Routing in the GUI. See “[Routing Clock Nets](#)” on page 6-72 for more information.
2. Analyze the clock mesh circuitry by using the `analyze_subcircuit` command. See “[Analyzing Clock Mesh Circuits](#)” on page 5-118 for more information.
3. Analyze the results for the complete multisource clock tree. See “[Analyzing the Clock Tree Results](#)” on page 5-134 for more information.

H-Tree-Based Multisource Clock Trees

Instead of a mesh, the multisource drivers can also be driven by an H-tree. An H-tree structure has a higher tolerance to cross-corner variation due to its uniform structure and use of identical cells.

To build an H-tree-based multisource clock tree, carry out the following tasks:

- Add the multisource drivers. See “[Inserting Multisource Drivers](#)” on page 5-126 for more information.
 - Build and route the H-tree. See “[Creating H-, T-, or I-Shape Routes for Premesh Trees](#)” on page 5-115 for more information.
 - Build the subtrees. See “[Building the Subtrees](#)” on page 5-132 for more information.
 - Route the clock nets and analyze the results. See “[Routing Clock Nets and Analyzing Results](#)” on page 5-133 for more information.
-

Analyzing the Clock Tree Results

After synthesizing the clock trees, analyze the results to verify that they meet your requirements. Typically the analysis process consists of the following tasks:

- Analyzing the clock tree reports (as described in “[Generating Clock Tree Reports](#)” on page 5-135)
- Analyzing the clock tree timing (as described in “[Analyzing Clock Timing](#)” on page 5-139)
- Reporting QoR (as described in “[Generating Categorized Timing Reports for Clock Tree Synthesis](#)” on page 5-140)
- Verifying the placement of the clock instances, using the GUI (as described in “[Analyzing Clock Trees in the GUI](#)” on page 5-143)

If the clock trees meet your requirements, you are ready to analyze the entire design for quality of results, as explained in “[Analyzing and Refining the Design](#)” on page 5-167.

If the synthesis results do not meet your requirements, the tool can help you debug the results by outputting additional information during clock tree synthesis. Set the `cts_use_debug_mode` variable to `true` before running clock tree synthesis to output the following additional information:

- User-defined design rule constraints (maximum transition time, maximum capacitance, and maximum fanout)
For more information about user-defined design rule constraints, see “[Setting Clock Tree Design Rule Constraints](#)” on page 5-50.
- User-defined clock tree timing constraints (skew and insertion delay)
For more information about user-defined clock tree timing goals, see “[Setting Clock Tree Timing Goals](#)” on page 5-53.
- Clustering targets set by the tool (maximum transition time and maximum capacitance)

In addition, you can output detailed characterization data for the clock tree references by setting the `cts_do_characterization` variable to `true`.

Using this information, you can change the clock tree definitions to improve your results.

Generating Clock Tree Reports

To generate clock tree reports, use the `report_clock_tree` command or choose Clock > Report Clock Tree in the GUI. By default, a report is generated for all clock trees for the current scenario. To limit the report to specific clock trees for the current scenario, use the `-clock_trees` option. To generate the report for a specific operating condition, use the `-operating_condition` option.

To generate the report for all clock trees of specific scenarios, use the `-scenarios` option.

Note:

The `-clock_trees` option and the `-scenarios` option are mutually exclusive; you can specify only one of these options.

You can use the `-histogram_transition`, `-histogram_capacitance`, `-histogram_rcdelay`, `-histogram_rcdelay_to_sink`, and `-histogram_fanout` options to report clock tree transition, capacitance, RC values, and fanout individually in a file in histogram-style format. When you specify the histogram options, use the `-number_of_bins` option to indicate the number of histogram bins.

[Table 5-17](#) shows the clock tree reports that are supported by IC Compiler.

Table 5-17 Clock Tree Reports

Report type	Command option (GUI object)	Description
Default		Reports the global skew for the specified clock trees.
Clock path from specific roots	-from ("From pins" check box)	Reports the fanout clock trees from the specified pins.
Clock path to specific sinks	-to ("To pins" check box)	Reports the clock paths to the specified sink pins.
DRC violators	-drc_violators ("DRC violators" check box)	Lists the design rule violations that occur in the clock trees up to the endpoints (default sinks or don't touch subtrees), including violations beyond exception on stop pins, exclude pins, and float pins but excluding violations on don't buffer nets, don't touch subtrees, nets inside block abstraction models, and nets connected to boundary cells or pad cells.
All DRC violators	-all_drc_violators ("Include violators beyond exceptional pins" check box)	Lists all design rule violations that occur in the clock trees up to the default sinks through don't touch subtrees, including violations beyond exception pins.
Exceptions	-exceptions ("Exceptions" check box)	Lists the stop pins, exclude pins, float pins, nonstop pins, don't touch subtrees, don't buffer nets, don't size cells, and size-only cells.
Level information	-level_info ("Level information" check box)	Reports on the structural and timing characteristics for all levels of the clock trees.
Settings	-settings ("Settings" check box)	<p>Lists the following global clock tree settings: clock tree synthesis, logic-level balancing, clock tree design rule constraints, clock tree configuration files nondefault routing rules, OCV clustering.</p> <p>Lists the following per-clock tree settings: clock tree timing constraints, clock tree design rule constraints, maximum buffer levels, clock tree optimizations.</p>

Table 5-17 Clock Tree Reports (Continued)

Report type	Command option (GUI object)	Description
Sinks	-show_all_sinks ("Show all default sinks in design" check box)	Lists the default sink pins.
Sink Groups	-sink_group ("Sink Group" check box)	Lists clock sink groups and their timing relationships. See " Analyzing Clock Sink Groups " on page 5-12 for details.
Structure	-structure ("Clock tree structure" option) -partial_structure_within_exceptions ("Clock tree structure within exceptions" option) -premesh ("Pre-mesh information" check box) -postmesh ("Post-mesh information" check box)	Lists the structure of the clock tree. Lists the structure of the clock tree up to any explicit exclude pins or float pins. The clock structure beyond these exception pins is not reported. Reports the clock network that has a clock mesh, starting from the clock root to the mesh driver inputs. Reports the clock network that has a clock mesh, starting from the inputs of the clock drivers that are the load pins of the clock mesh.
Summary	-summary ("Summary" check box)	Reports the global clock skew summary.

Table 5-17 Clock Tree Reports (Continued)

Report type	Command option (GUI object)	Description
Histogram Analysis	-histogram_transition ("Transition" check box)	Reports the transition of the selected clock trees in the specified file.
	-histogram_capacitance ("Capacitance" check box)	Reports the capacitance of the selected clock trees in the specified file.
	-histogram_rcdelay ("RC delay" check box)	Reports the RC values of the selected clock trees in the specified file.
	-histogram_rcdelay_to_sink ("RC delay to sink" check box)	Reports the RC delay values per sink between the clock drivers and each valid sink in the specified file.
	-histogram_fanout ("Fanout" check box)	Reports the fanout of the selected clock trees in the specified file.
	-number_of_bins ("Number of bins" check box)	Specifies the number of histogram bins. When you specify the histogram options, you can use this option to set the histogram bins. The default is 10.

Generating Reports for Multicorner-Multimode Designs

To generate a clock tree report for a particular scenario corner, set the corner by using the `set_scenario_options` command and then run the `report_clock_tree` command. For example, to report the clock trees for the scn1, scn2, and scn3 scenarios for the minimum corner, enter the following commands:

```
icc_shell> set_scenario_options -scenarios {scn1 scn2 scn3} \
    -cts_mode true -cts_corner min
icc_shell> report_clock_tree -scenarios {scn1 scn2 scn3} \
    -operating_condition min
```

Because the `report_clock_tree` command requires that you define clocks in the scenario that you want to report, you should define clocks in every scenario when using multicorner clock tree optimization.

```
icc_shell> create_scenario scn1
icc_shell> set_operating_conditions -max $opcond1
icc_shell> set_tlu_plus_files -max_tluplus $tlu1 -tech2itf_map $tlu_map
icc_shell> create_clock -name clock -period 4 clock

icc_shell> create_scenario scn2
icc_shell> set_operating_conditions -max $opcond2
icc_shell> set_tlu_plus_files -max_tluplus $tlu2 -tech2itf_map $tlu_map
icc_shell> create_clock -name clock -period 4 clock
```

Analyzing Clock Timing

The timing characteristics of the clock network are important in any high-performance design. To obtain detailed information about the clock networks in a design, use the `report_clock_timing` command. This command reports the clock latency, transition time, and skew characteristics at specified clock pins of sequential elements in the network.

In the `report_clock_timing` command, use the `-type` option to specify the type of report you want (see [Table 5-18](#) for available report types), the scope of the design to analyze, and any desired filtering or ordering options for the report. For analyzing the effect of clock tree synthesis exceptions and clock propagation, use the `-cts_mode` option. IC Compiler gathers the requested information and reports it in the specified order.

By default, the `report_clock_timing` command reports for the current scenarios only. Use the `-scenarios` option to report for the specified scenarios.

Note:

You cannot use the clock tree specification options `-clock`, `-from_clock`, and `-to_clock`, with the `-scenarios` option. The report has details of all clock trees of the specified scenarios.

Table 5-18 Clock Timing Report Types

To generate this report	Use this keyword
Single-clock local skew	skew
Transition	transition
Interclock skew	interclock_skew
Latency	latency
Summary (shows the worst instances of transition time, latency, and skew for the specified clocks)	summary

Generating Categorized Timing Reports for Clock Tree Synthesis

You can generate categorized timing reports to analyze the clock tree synthesis results for the design in its current state by using the `create_qor_snapshot` command (or by choosing Timing > Create QoR Snapshot in the GUI). This command measures and reports the quality of the design in terms of timing, design rules, area, power, congestion, routing, and so on. It stores the quality information in a set of snapshot files. You can retrieve and report the snapshot with the `report_qor_snapshot` command. You can also selectively retrieve, sort, and display the information from the snapshot with the `query_qor_snapshot` command (or by choosing Timing > Query QoR Snapshot in the GUI and then selecting CTS Mode in the Query Mode setting).

You can generate the quality of results snapshot for the current scenario only or for any specified set of scenarios.

create_qor_snapshot

The `create_qor_snapshot` command measures the quality of the design in its current state and stores the quality information into a set of report files. You can capture the quality information using different optimization strategies or at different stages of the design and compare the quality results. Use the `-clock_tree` option of the `create_qor_snapshot` command to generate the information for clock tree synthesis categorized timing reports. See “[“create_qor_snapshot” on page 4-63](#) for details.

query_qor_snapshot

The `query_qor_snapshot` command reads in a QoR report generated by previous usage of the `create_qor_snapshot` command and analyzes the results by collecting information about each path. You can query the collected information with various filters and display the

results in text or HTML format, or in an interactive HTML window linked to the GUI layout view. The interactive HTML report lets you quickly find paths with certain problems such as large fanouts or transition degradation, and then view the path in the current design.

This is the syntax of the `query_qor_snapshot` command for generating clock tree synthesis categorized timing reports:

```
query_qor_snapshot
  -clock_tree_only
  [-name snapshot_name]
  [-directory directory_name]
  [-cts_output_file file_name]
  [-cts_sort_by column_list]
  [-cts_columns column_list]
  [-cts_and column_list]
  [-cts_filters filter_list]
  [-cts_set_exceptions_file file_name]
  [-cts_clear_exceptions_file file_name]
```

Alternatively, you can create a QoR snapshot in the GUI by choosing Timing > Query QoR Snapshot and then selecting CTS Mode in the Query Mode setting.

The command options let you filter and sort the query results and display those results in text or HTML format. The command reports a summary of the clock tree, subject to one or more specified filters by using the `-cts_filters` option. All values that match a filter are highlighted (with a * for text or red text for HTML). You can specify the columns to display with the `-cts_columns` option, the column value sorting order with the `-cts_sort_by` option, and the filters to apply as AND filters with the `-cts_and` option. Use the `-cts_set_exceptions_file` option to generate a Tcl command file containing `set_clock_tree_exceptions` commands for each clock path matching one or more filters that can create clock tree exceptions. Use the `-cts_clear_exceptions_file` option to generate a Tcl command file containing `remove_clock_tree_exceptions` commands for each clock path matching one or more filters that can remove clock tree exceptions.

If you do not specify the `-cts_filters` option, the `query_qor_snapshot` command automatically applies a set of default filters. The default filter settings are shown in the following example:

```
icc_shell> query_qor_snapshot -clock_tree_only -name clk_report \
  -cts_filters {-transitionViolation 1 -fanoutViolation 1 \
  -capacitanceViolation 1 -dontTouchSubtree 1 \
  -dontBufferNets 1 -dontSizeCells 1 -sizeOnlyCells 1 \
  -explicitIgnore 1 -skew 0 \
  -transitionDegradation 2.0 -slewDegradation 2.0}
```

The `query_qor_snapshot` command does not perform any additional analysis of the design, but merely processes the report information already saved in the snapshot report files. Therefore, execution of the `query_qor_snapshot` command is much faster than execution of the `create_qor_snapshot` command.

The following example looks at all clock paths matching the ETM impossible pins filter. An ETM impossible pin is a clock sink pin inside an extracted timing model (ETM) that has zero or negative latency. If the clock path has one or more ETM impossible pins, the command automatically generates the appropriate `set_clock_tree_exceptions` commands for those clock paths.

```
icc_shell> query_qor_snapshot -clock_tree_only -name clk_report ... \
    -cts_filters {-etm_impossible_pins 1} -cts_set_exceptions_file {my_}
```

The following example looks at all clock paths matching the don't touch subtree filter. If the clock path has one or more don't touch subtree exceptions, the command automatically generates the appropriate `remove_clock_tree_exceptions` commands for those clock paths.

```
icc_shell> query_qor_snapshot -clock_tree_only -name clk_report ... \
    -cts_filters {-dont_touch_subtree 1} -cts_clear_exceptions_file {my_}
```

In the following example, the `query_qor_snapshot` command reports all clocks with skew greater than 5 ns:

```
icc_shell> query_qor_snapshot -clock_tree_only -name clk_report ... \
    -cts_filters {-skew 5.0}
```

In the following example, the `query_qor_snapshot` command reports all clocks that have cells with an output transition value or input transition value over 2:

```
icc_shell> query_qor_snapshot -clock_tree_only -name clk_report ... \
    -cts_filters {-transition_degradation 2}
```

In the following example, the `query_qor_snapshot` command reports all clocks having ETM impossible pins and sets the ETM impossible pin threshold to 0.1 ns:

```
icc_shell> query_qor_snapshot -clock_tree_only -name clk_report ... \
    -cts_filters {-etm_impossible_pins 1 -etm_impossible_threshold 0.1}
```

In the following example, the `query_qor_snapshot` command reports all clocks that have path segments with slew degradation over 3:

```
icc_shell> query_qor_snapshot -clock_tree_only -name clk_report ... \
    -cts_filters {-slew_degradation 3}
```

In the following example, the `query_qor_snapshot` command reports all clocks that have over 50 logic levels:

```
icc_shell> query_qor_snapshot -clock_tree_only -name clk_report ... \
    -cts_filters {-logic_level 50}
```

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.

When you specify the `-congestion` option with the `clock_opt_feasibility` command, the command performs placement to meet the congestion optimization requirements. By default, the command considers scan nets in the design during congestion estimation. To specify to continue placement when the design contains scan chains but no SCANDEF data, specify the `-continue_on_missing_scandef` option.

Analyzing Clock Trees in the GUI

The IC Compiler GUI provides tools in the layout window and the interactive clock tree synthesis (CTS) window to perform both high-level and detailed analyses of the clock tree structures and timing in your design.

- In the layout window, the clock tree visual modes allow you to view the overall structures and timing distribution of the clock trees.
- In the interactive clock tree synthesis window, you can view information about all the clocks in the design and detailed information about individual clock trees.

You can select clock tree objects in the interactive clock tree synthesis window to highlight them in the layout view or color them in the clock tree visual mode. For information about layout view visual modes, see [“Using Visual Modes” on page A-72](#).

To analyze clock tree timing, you can

- Examine the distribution of clock tree latency or transition times by using the clock tree latency/transition visual mode in a layout view
- View the distribution of clock tree pin arrival times in a clock tree arrival histogram
- Examine the fanin and fanout network for selected clock tree structures in clock tree schematics
- Highlight the maximum or minimum fanout path through a clock tree in the clock tree fanout browser or a clock tree schematic
- Examine clock tree latency over time for individual clock trees in clock tree latency graph views

To analyze clock tree structures, you can

- Examine the clock trees, their levels, or their objects by using the clock trees visual mode in a layout view
- View or change clock tree options in the clock tree options viewer
- View clock details and select clocks on the Clock Browser panel
- Examine details about the fanout network for a clock tree in the clock tree fanout browser
- Examine the fanout structure of clock networks before and after clock tree synthesis in clock tree levelized graph views
- Find and view information about specific clock tree objects (cells, pins, or load nets) in clock tree synthesis object list views

For information about these subjects, see the following sections:

- [Viewing Clock Trees in the Layout Window](#)
- [Viewing and Setting Clock Tree Options](#)
- [Using the Interactive Clock Tree Synthesis Window](#)
- [Viewing the Distribution of Clock Arrival Times](#)
- [Exploring Clock Tree Structures](#)
- [Examining Fanout or Fanin Paths in the Clock Tree Network](#)
- [Highlighting the Critical Paths](#)

- [Viewing Abstract Clock Tree Graphs](#)
- [Examining Clock Tree Latency](#)
- [Examining Clock Tree Fanout Levels](#)
- [Examining Clock Tree Overlaps](#)
- [Listing Clock Tree Objects](#)

Viewing Clock Trees in the Layout Window

The IC Compiler tool provides two clock tree visual modes:

- Clock tree structure
- Clock tree timing distribution

These visual modes are useful for analyzing the clock tree structure after clock tree synthesis. Analyzing the structure enables you to identify possible sources for clock tree timing problems, such as a restrictive floorplan or the use of nonoptimal references.

Viewing the Clock Tree Structure

To display the clock tree structure in the active layout view,

1. Choose Clock > Color By Clock Trees.
The Visual Mode panel appears.
2. Click Reload.
The Configure Clock Tree Visual Mode dialog box appears.
3. Select the clock tree levels or cell types that you want to display.
To view the display choices for a clock tree, click the expand button (+) to the left of the clock tree name. You must select at least one level or cell type.
 - To display the entire clock tree in a single color, select AllLevel.
 - To display each selected clock tree level in a different color, select Level 0...Level *n*.
 - To display the buffers inserted into the clock tree by the tool, select buffer.
 - To display the inverters inserted into the clock tree by the tool, select inverter.
 - To display the preexisting gates in the clock tree, select preexisting.
 - To display the clock tree sinks, select sink.

To analyze the synthesized clock tree, you typically select one or more clock tree levels (Level 0...Level *n*) that you want to display. You can use this display to debug the placement of the clock tree cells.

4. Click OK or Apply.

The GUI displays the selected clock tree levels and cell types in the layout view.

By default, the clock tree nets are displayed. To display the clock tree cells, deselect Cells in the “Exclude objects” section. To disable the display of clock tree nets, select Nets in the “Exclude objects” section.

For more details, see the “Coloring Clock Trees” topic in IC Compiler Help.

Viewing the Clock Tree Timing Distribution

To display the clock latency or transition time distribution in the active layout view,

1. Choose Clock > Color By Clock Latency/Transition.

The Visual Mode panel appears.

2. Click Reload.

The Clock Latency/Transition dialog box appears.

3. In the Clock Latency/Transition dialog box, do the following:

- Type or select a clock tree in the Clock box.
- Select a Delay Type option (Latency or Transition).
- Set other options as needed.

4. Click OK.

For more details, see the “Coloring Clock Tree Latency and Transition” topic in IC Compiler Help.

Viewing and Setting Clock Tree Options

You can use the clock tree options viewer to view or set clock tree options for each clock tree in the design. The clock tree options viewer is available in both the layout window and the interactive clock tree synthesis window.

To open the clock tree options viewer, choose Clock > List Clock Tree Options.

The clock tree options viewer displays a list of all the clock trees in the design. The list contains columns with the current values (whether default or previously set) for each clock tree option. You can use the buttons below the list to set or change options for the selected clock tree, display or hide columns in the list, or close the dialog box.

For more information about the clock tree options viewer, see the “Viewing and Setting Clock Tree Options” topic in IC Compiler Help.

Using the Interactive Clock Tree Synthesis Window

The interactive clock tree synthesis window provides information about all of the clocks defined in your design. You can

- View clock details and select clocks on the Clock Browser panel
- View a high-level overview of the clock tree fanout timing in a clock tree arrival histogram
- Explore the pin-to-pin fanout structure of a clock tree in the clock tree fanout browser
- Examine the fanin or fanout network for an entire clock tree or for selected clock tree structures in clock tree schematics
- Highlight the critical path in a clock tree fanout browser or a clock tree schematic
- Examine clock latency and skew over time in a clock tree latency graph
- Examine the clock tree fanout structure in a clock tree leveled graph
- View a matrix of overlapping clock pairs in the clock tree overlap analyzer
- View object information in clock tree synthesis object list views

Clock arrival histograms provide a high-level overview of the timing quality in the fanout network of a selected clock tree. The Clock Browser panel and the clock tree fanout browser are the starting points for detailed clock tree analysis. You can open one clock browser in an interactive clock tree synthesis window, and you can open one clock tree fanout browser for each clock tree in the design.

To open an interactive clock tree synthesis window,

- Choose Clock > New Interactive CTS Window.

When you open an interactive clock tree synthesis window, the Clock Browser panel appears at the top of the window. The Clock Browser panel displays a table with information about the clock in your design. The table columns show the clock names, sources, and other details about each clock.

For more information about the interactive clock tree synthesis window and the Clock Browser panel, see the “Opening an Interactive CTS Window” and “Using the Clock Browser” topics in IC Compiler Help.

Viewing the Distribution of Clock Arrival Times

Clock arrival histograms provide a high-level overview of the timing quality in the fanout network of a selected clock tree. Create a clock arrival histogram to identify clock network pins that failed their constraints.

To display the clock arrival histogram,

- Choose Timing > Clock Arrival Histogram.

The clock arrival histogram appears in a new histogram view window. The window is split into two panes, with the histogram bar graph in the left pane and an object table in the right pane. The number at the top of the tallest bin indicates the number of clock tree objects in the bin. Green bins (on the positive side of 0) contain objects in the design that met their constraints. Red bins (on the negative side of 0) contain objects that failed their constraints.

You can select one or more pin names in the table for further examination with other analysis tools. You can

- Select pins and display the clock fanin network in a path schematic
- View object properties for the selected pins

For more information about clock arrival histograms, see the “Opening a Clock Tree Arrival Time Histogram” topic in IC Compiler Help.

Exploring Clock Tree Structures

You can use the clock tree fanout browser to display pin-to-pin fanout information about clock tree structures in the physical design. The clock tree fanout browser provides a hierarchical listing of the clock tree, which can be useful for analyzing complex clock tree structures before or after clock tree synthesis.

- Before you perform clock tree synthesis, use the clock tree fanout browser to understand the existing clock tree structure.
- After you perform clock tree synthesis, use the clock tree fanout browser to analyze the resulting clock tree structures and for troubleshooting if you failed to achieve the expected QoR.

If you are not familiar with the clock trees in your design, you can explore the clock tree structures and gather information about objects (buffers, inverters, and preexisting cells) at each level. You can use the clock tree fanout browser together with clock network fanin and fanout schematics or the clock trees visual mode to determine what constraints or options you need to modify to improve the clock tree synthesis results.

You can open one clock tree fanout browser view window for each clock tree.

To open the clock tree fanout browser for a clock tree,

1. Select the clock on the Clock Browser panel.
2. Choose View > Clock Tree Hierarchy Browser.

The view window consists of two panes, with an expandable level tree for each clock tree in the left pane and an object information table in the right pane. You can explore the complete structure of a clock tree, observe how many levels are present, and examine the clock tree fanout information for the cells at each level.

The object table shows clock tree fanout information for the clock tree objects you select in a level tree. You can display information about child (next level) objects or associated objects by selecting an option in the list above the object table. If you hold the pointer over a clock tree object, in either the left or right pane of the clock tree hierarchy browser, information about the object appears in an InfoTip.

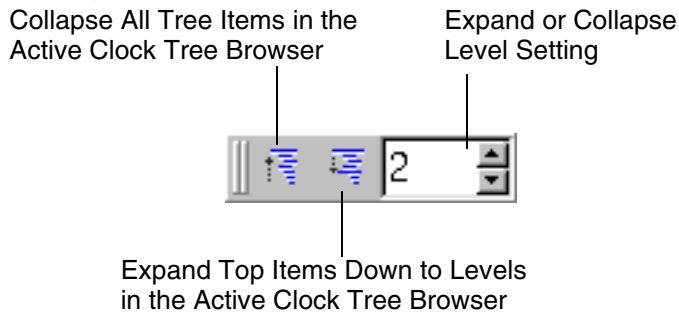
For more information about using the clock tree fanout browser, see the following sections:

- [Expanding or Collapsing Fanout Levels](#)
- [Highlighting a Selected Object](#)
- [Highlighting the Critical Paths](#)
- [Changing the Timing Mode](#)

Expanding or Collapsing Fanout Levels

The Expand Browser toolbar, shown in [Figure 5-33](#), lets you expand or collapse fanout levels for the clock tree displayed in the active clock tree fanout browser. You can expand or collapse all levels, or expand a specified number of levels.

Figure 5-33 Expand Browser Toolbar

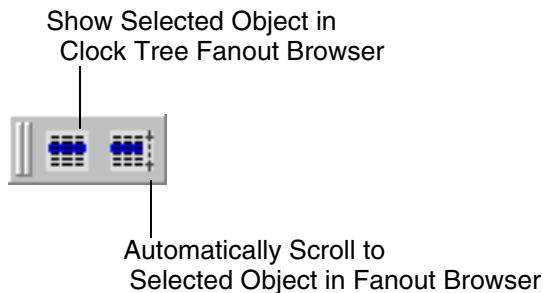


Highlighting a Selected Object

You can locate a specific clock tree object in the fanout browser by selecting the object in another view, such as a layout or schematic view. You can also enable a mechanism in a clock tree fanout browser that automatically displays and highlights an object when you select it in another view.

The Follow Selection toolbar, shown in [Figure 5-34](#), lets you find a clock tree object in the clock tree fanout browser by selecting the object in another view.

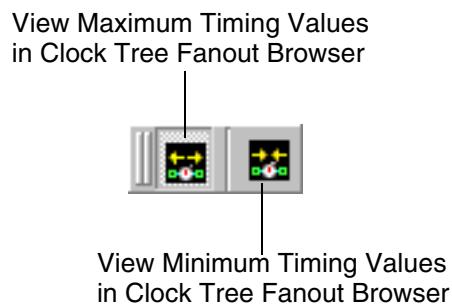
Figure 5-34 Follow Selection Toolbar



Changing the Timing Mode

When you view object information in a clock tree fanout browser, you use the Clock Tree Timing Modes toolbar, shown in [Figure 5-35](#), to control whether the tool uses maximum or minimum timing values to calculate the timing data.

Figure 5-35 Clock Tree Timing Modes Toolbar



Examining Fanout or Fanin Paths in the Clock Tree Network

You can use clock tree schematics to examine timing problems by focusing on critical fanin or fanout paths in the clock tree network. A clock tree schematic displays the structure of selected clocks in a flat, single-sheet schematic of cells and nets that have been placed and routed with a general purpose. You can generate a clock tree schematic to display an entire clock path or the fanin or fanout logic for selected clock tree objects.

To display clock paths in a clock tree schematic,

1. Select the clocks on the Clock Browser panel.
2. Choose Schematic > New Clock Fanout Schematic > Of Selected Clocks.

To display the fanin or fanout logic for selected clock tree objects,

1. Select the objects in the clock tree fanout browser.
2. Choose one of the following commands:
 - To display fanin logic, choose Schematic > New Clock Fanin Schematic of Selected Pins/Ports.
 - To display fanout logic, choose Schematic > New Clock Fanout Schematic > Of Selected Pins/Ports.

You can add logic or timing paths in a clock tree schematic by using any of the following methods:

- Select an object and add fanin or fanout logic for the object
- Select an object and add timing paths with the worst slack from, through, or to the object
- Add timing paths with the worst slack in the design or in a path group

For details about adding logic or timing paths in schematic views, see the “Adding Logic or Timing Paths in a Path Schematic” topic in IC Compiler Help.

You can expand or collapse cell or net levels in a clock tree schematic. When you open the schematic, all cell and net levels are initially collapsed.

To expand levels for selected cells or nets,

1. Select one or more cells, nets, or both.
2. Right-click and choose one of the following commands:
 - Expand Selected
 - Expand First Level
 - Expand Last Level

To collapse levels for selected cells or nets,

1. Select one or more cells, nets, or both.
2. Right-click and choose Collapse Selected.

To expand or collapse levels for all unselected cells and nets in the schematic,

1. Select any cells or nets you do not want to expand or collapse.

If you want to quickly expand or collapse all the cell and net levels in the schematic, click in the background to deselect all selected objects.

2. Right-click and choose one of the following commands:
 - Expand Unselected
 - Collapse Unselected

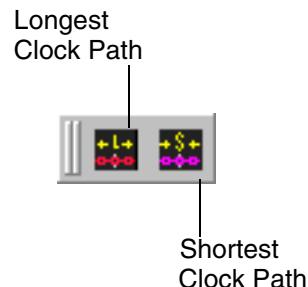
Clock-gating cells are generally not of interest when you explore a clock tree schematic, and they prevent you from fully collapsing the clock trees when you analyze a clock-gated netlist. You can select an option in the Application Preferences dialog box to collapse clock-gating cells automatically when you open a clock tree schematic. For more details, see “[Setting GUI Preferences](#)” on page A-22.

For more information about clock tree schematics, see the “Viewing Clock Tree Schematics” topic in IC Compiler Help.

Highlighting the Critical Paths

You can use the Critical Path toolbar, shown in [Figure 5-36](#), to highlight the longest and shortest clock tree fanout paths in the active clock tree schematic or clock tree fanout browser.

Figure 5-36 Critical Path Toolbar



For more information, see the “Highlighting the Critical Fanout Paths” topic in IC Compiler Help.

Viewing Abstract Clock Tree Graphs

Abstract clock graph views are similar to clock tree schematics but with a higher level of abstraction. You can focus on key characteristics of the clock trees while ignoring irrelevant objects.

To display an abstract clock graph for all the clock trees in your design in the interactive clock tree synthesis window:

- Choose View > Abstract Clock Graph From All

To display an abstract clock graph for one or more clock trees:

1. Select the clocks on the Clock Browser panel.
2. Choose View > Abstract Clock Graph From Selected.

An abstract clock graph view displays clock tree objects by using symbols and arcs.

- Symbols represent visible objects, such as clocks, clock tree exceptions, clock gates, buffers, inverters, and sequential cells.
- Arcs represent the clock paths between visible objects and are displayed as flylines.

For more details, see the “Viewing Abstract Clock Tree Graphs” topic in IC Compiler Online Help.

Examining Clock Tree Latency

You can use a latency abstract clock graph to examine clock skew, insertion delay, and bottlenecks in a clock tree. The latency abstract clock graph represents the time-based scheduling relationship of the clock network that facilitates clock tree latency and skew analysis. The latency abstract clock graph does not show details about logic connectivity.

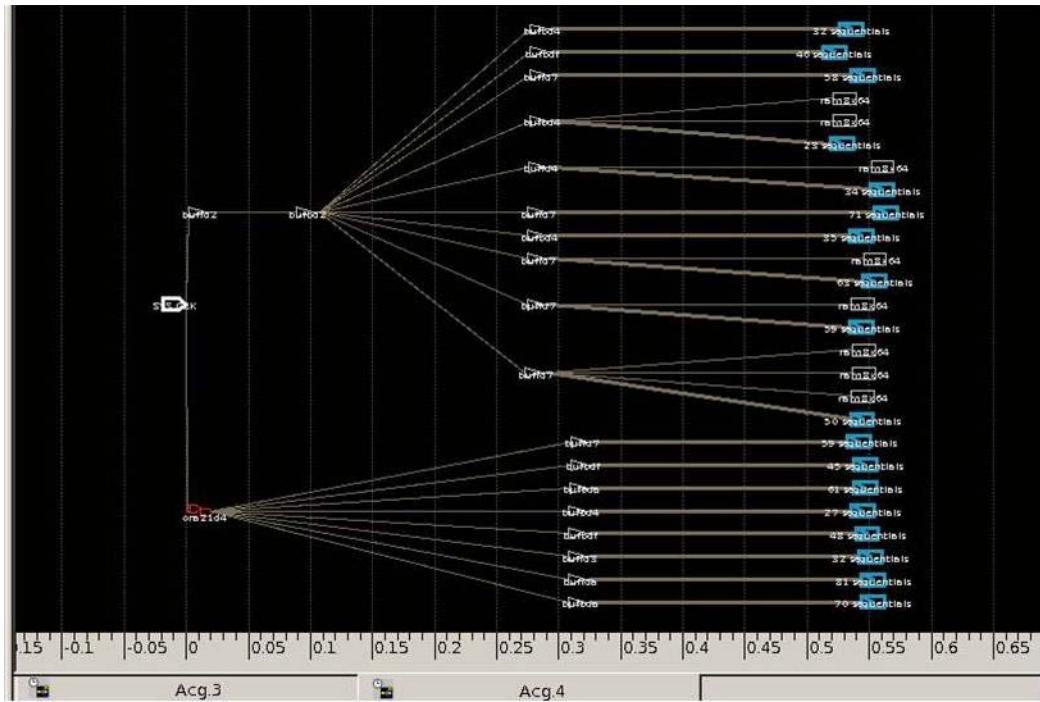
Note:

The latency abstract clock graph is available only after you have performed clock tree synthesis.

To display clock tree latency for one or more clock trees in a latency abstract clock graph view:

1. Select the clocks on the Clock Browser panel.
2. Choose View > Latency Abstract Clock Graph From Selected.

[Figure 5-37](#) shows a latency abstract clock graph. The x-axis represents clock tree logic relative to time. The graph positions logic gates based on the latency of their input clock pins. For more details, see the “Examining Clock Tree Latency Over Time” topic in IC Compiler Online Help.

Figure 5-37 Latency Abstract Clock Graph

Examining Clock Tree Fanout Levels

You can use a leveled abstract clock graph to visualize the relationships between driving cells and output loads in a clock network or the relationships between clock trees. The leveled abstract clock graph represents the logic gates based on the levels of their clock inputs in the clock tree fanout. The leveled abstract clock graph does not show details about logic connectivity.

Note:

The leveled abstract clock graph is available both before and after you perform clock tree synthesis.

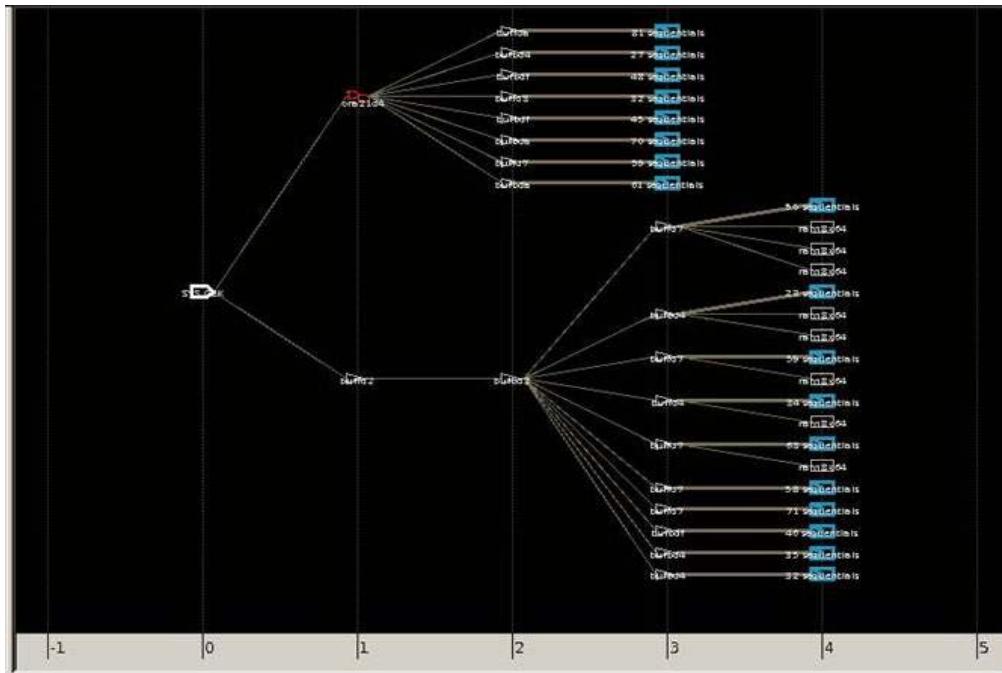
To display the entire clock fanout for one or more clocks in a leveled abstract clock graph view,

1. Select the clocks on the Clock Browser panel.
2. Choose View > Levelized Abstract Clock Graph From Selected.

[Figure 5-38](#) shows a leveled abstract clock graph. The x-axis represents clock tree fanout levels. The graph positions logic gates based on the fanout levels of their clock input pins. The fanout level of a reconvergent point is the maximum level of the fanout branches through

that point. For more details, see the “Examining Clock Tree Fanout Levels” topic in IC Compiler Online Help.

Figure 5-38 Levelized Abstract Clock Graph



Examining Clock Tree Overlaps

The clock tree overlap analyzer view displays a hierarchical or flat list of clocks and a matrix of overlapping clock pairs. The overlap matrix consists of a grid with a row and column for each clock in the clock list.

To open the clock tree overlap analyzer,

- Choose View > Clock Tree Overlay Analyzer.

In the clock list, expandable hierarchy trees represent the clock hierarchy among primary and generated clocks. You can click the expansion button (plus sign) next to a clock name to expand the tree, showing the names of the clocks at the next level in the hierarchy.

The overlap matrix consists of a grid with a row and column for each clock in the clock list. The Index column on the left side of the matrix correlates the clock names in the list with the rows and columns in the matrix. A light gray color in the grid indicates that clocks belong to the same clock domain. When you select a clock name in the clock list, a light blue color highlights the row for the corresponding clock in the matrix.

The matrix displays X symbols in the grid to indicate overlapping clock pairs. If you click a box in the grid that represents an overlap, the tool displays the overlap point in the selection color, which is white by default, in other views such as a clock tree schematic or clock graph view.

For more information about the clock tree overlap analyzer, see the “Examining Clock Tree Overlaps” topic in IC Compiler Help.

Listing Clock Tree Objects

You can search for objects in a clock tree and view detailed information about them in an object tree list view. You can customize the list, and you can select objects in the list for further examination with other analysis tools.

The object table list view consists of an object table and a button bar.

- The table displays details about the objects in the list.
- The button bar provides buttons you can use to manage the table contents and access other analysis tools.

You can customize the table columns, save the object details in a text file, or reload the table with details for different objects. You can also access analysis tools such as histograms.

To find and list clock tree objects,

1. Choose Edit > Find CTS Objects By.
The Find CTS Object dialog box appears.
2. Select the name of the clock tree you want to search.
3. Set search criteria options as needed.

You can search for objects of a particular type or search a particular number of levels. You can also filter the objects by specifying a name or name pattern.

4. Select the output option for the type of objects you want to display in the list.
The choices are Cells, Pins, and “Load nets.” The default is Pins.
5. Click OK.

The tool finds the clock tree objects that meet the search criteria and displays their names and other information in a clock tree synthesis object list view.

You can select objects in the list by clicking or dragging the pointer across their names. You can customize the object information that appears in the table by displaying, hiding, or reorganizing the table columns.

When you select an object in the table, you can click buttons on the button bar to

- Display the object in the clock tree fanout browser
- Display the object in a clock tree schematic
- Highlight one or more objects in the layout view

You can also generate a histogram that shows the distribution of values in a table column.

For more information, see the “Listing Clock Tree Objects” topic in IC Compiler Help.

Fine-Tuning the Clock Tree Synthesis Results

The IC Compiler tool supports the following methods for fine-tuning the clock tree synthesis results:

- [Using the Useful Skew Technique](#)
- [Balancing the Skew Using Skew Groups](#)
- [Resynthesizing the Clock Trees](#)
- [Modifying the Nondefault Routing Rule](#)
- [Modifying Clock Trees in the GUI](#)

Using the Useful Skew Technique

The useful skew technique improves timing QoR by adjusting the clock arrival times to take advantage of positive slack in the network. Use the `skew_opt` command to automatically perform useful skew analysis and generate the useful skew constraints.

skew_opt Flow

The flow for using the `skew_opt` command is

1. Determine the existing clock latencies by doing a quick `clock_opt` run.

Use the `set_latency_adjustment_options` command to identify the timing relationships between a source clock and associated generated clocks or virtual clocks. For example, for a generated clock, enter

```
icc_shell> set_latency_adjustment_options \
           -from_clock master_clk -to_clock gen_clock
```

Use the `skew_opt -clock_balancing_only` to configure interclock delay balancing groups. The generated `skew_opt` script file includes the annotation of the changed ideal

latencies for clock objects. In addition, to ensure the same I/O latencies that are seen by `skew_opt` and the `psynopt` command that is embedded in `clock_opt`, you should perform clock routing separately from the `clock_opt` command to avoid an additional `update_clock_latency` call inside `clock_opt` after clock routing.

The following script shows an example of how to determine the latencies for your design:

```
icc_shell> place_opt
icc_shell> save_mw_cel -as placed
icc_shell> close_mw_cel
icc_shell> open_mw_cel placed
icc_shell> set_latency_adjustment_options ...
icc_shell> skew_opt -clock_balancing_only
icc_shell> clock_opt -inter_clock_balance \
    -update_clock_latency -no_clock_route
icc_shell> route_zrt_group -all_clock_nets \
    -reuse_existing_global_route true
icc_shell> extract_rc
```

2. Run the `skew_opt` command.

```
icc_shell> set_app_var skew_opt_optimize_clock_gates true
icc_shell> set_app_var skew_opt_optimize_buffer_count true
icc_shell> skew_opt
icc_shell> close_mw_cel
```

Note:

When you run the `skew_opt` command, set the `skew_opt_optimize_clock_gates` and the `skew_opt_optimize_buffer_count` variables to `true` to enable the optimization of clock gates and the reduction of buffers.

3. Load the updated clock latencies and useful skew solution.

```
icc_shell> open_mw_cel placed
icc_shell> set_latency_adjustment_options ...
icc_shell> source skew_opt.tcl
```

4. Perform clock tree synthesis and optimization with interclock delay balancing.

```
icc_shell> clock_opt -inter_clock_balance -no_clock_route
icc_shell> route_zrt_group -all_clock_nets \
    -reuse_existing_global_route true
icc_shell> extract_rc
icc_shell> route_opt
```

Important:

Because the `skew_opt` command works across all clock domains in your design, you must perform interclock delay balancing when running clock tree synthesis after `skew_opt`.

In addition, do not update the clock latency after clock tree synthesis (`clock_opt -update_clock_latency` option or `update_clock_latency` command). Updating

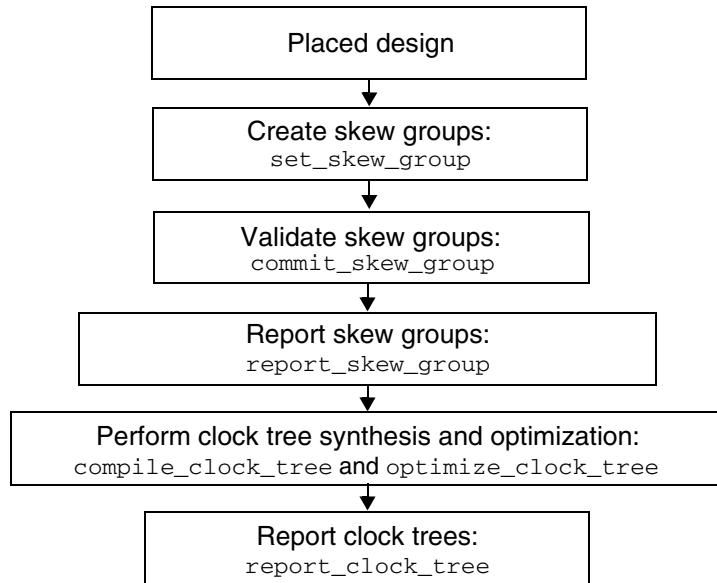
the clock latency after clock tree synthesis invalidates the script file generated by the `skew_opt` command.

Balancing the Skew Using Skew Groups

You can select a group of clock sinks from the same clock domain to form a subgroup named a skew group. The tool automatically balances the skew of clock sinks, including the clock pins of integrated clock-gating cells, enable flip-flops, and clock dividers, in each skew group. You can specify different skew and insertion delay constraints for different skew groups in the same clock domain.

[Figure 5-39](#) shows the flow for defining skew groups.

Figure 5-39 Skew Group Flow



Guidelines for Defining Skew Groups

When defining a skew group, use the following guidelines:

- Select nonhierarchical pins such as leaf stop pins, leaf float pins, the clock pins of integrated clock-gating cells, and the clock pins that start the sequential timing arcs.
- Select pins from the same master-clock domain only. Pins from different generated clock domains are allowed in the same skew group.
- Each pin can only belong to one skew group.
- No pins should have upstream-downstream dependency in the same skew group.

- A skew group that contains any nonstop pins or clock pins of integrated clock-gating cells creates dependency on the skew group to which their downstream sink pins belong.
- Clock sinks that are not defined in any skew groups are automatically included in the default skew group.
- A skew group is independent when only clock sink pins are defined in the group.
- No dependency loop between skew groups is allowed.

For example, if skew group A is dependent on skew group B, then skew group B cannot be dependent on other skew groups.

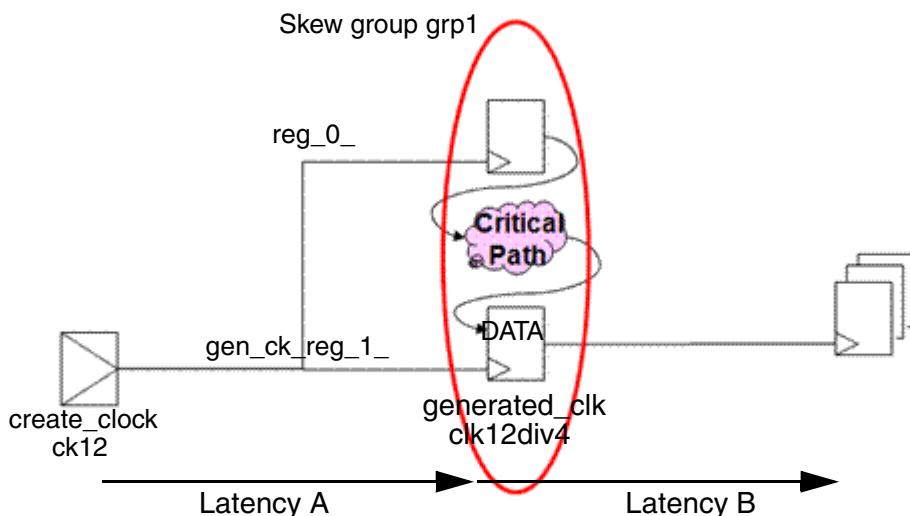
The guidelines also apply to the default skew group.

To create a skew group, use the `set_skew_group` command or choose Clock > Skew Group > Set Skew Group in the GUI.

For example, the following command creates a skew group named grp1 on the timing-critical path from `reg_0_/CLK` to `gen_ck_reg_1_/DATA` with two clock sinks, `reg_0_/CLK` and `gen_ck_reg_1_/CLK`, as shown in [Figure 5-40](#).

```
icc_shell> set_skew_group -name grp1 {reg_0_/CLK gen_ck_reg_1_/CLK}
```

Figure 5-40 Skew Group grp1 With Two Clock Sinks



The skew group grp1 automatically balances the skew, so you do not need to either adjust the clock latencies, latency A and latency B, or specify the float pin value at `reg_0_/CLK` to avoid setup violations.

You can specify the skew and insertion delay constraints by using the `-target_skew` and `-target_early_delay` options respectively. Note that the skew group constraints take precedence over the clock tree design rule constraints and the design rule constraints of the

logic library and the design. If you do not specify the skew or insertion delay constraints for a skew group, the skew group inherits the most strict clock tree design rule constraints of the clock sink in the clock domain. The default skew group always uses the clock tree design rule constraints.

To report the skew of each skew group, use the `report_clock_tree -skew_group` command.

Validating and Committing Skew Groups

After creating or modifying a skew group, you should always commit the skew group by using the `commit_skew_group` command or choose Clock > Skew Group > Commit Skew Group in the GUI. The command checks the skew group in the design against the guidelines. When the command detects an invalid skew group, it generates an error message and makes no changes to the netlist. If all skew groups are valid, the command restructures the clock trees to separate the skew groups. When you specify the `-check_only` option, the command only validates the skew groups, but it does not restructure the clock trees.

You can report the settings of skew groups by using the `report_skew_group` command or by choosing the Clock > Skew Group > Report Skew Group in the GUI.

To remove a skew group, use the `remove_skew_group` command or choose Clock > Skew Group > Remove Skew Group in the GUI

Limitations of Using Skew Groups

Using skew groups has the following three limitations:

- If you run the `remove_clock_tree` command on a design that contains skew groups, you must rerun the `commit_skew_group` command before balancing the clock trees again.
- If there is dependency between skew groups, the `-target_skew` and `-target_early_delay` options are not honored.
- You cannot use skew groups with features such as clock tree configuration files, logic-level balancing, and OCV clustering.

Resynthesizing the Clock Trees

In some cases, to achieve the best results, you must fine-tune the clock tree synthesis settings, then resynthesize the clock trees.

Resynthesizing the clock trees consists of the following steps:

1. Input the postplacement design that you saved before performing clock tree synthesis.
If you did not save the design before running clock tree synthesis (or if you want to keep some synthesized clock trees, but resynthesize others), you must remove the synthesized clock trees before you can resynthesize the clock trees (see “[Removing Clock Trees](#)” on page 5-80). The resulting design might not be identical to your design before running the initial clock tree synthesis, because some gates might have been moved or sized during the clock tree synthesis process.
2. Validate the setup.
3. Fine-tune the clock tree synthesis settings.
4. Resynthesize the clock trees.

The following sections describe these steps.

Validating the Setup

If the synthesis results do not meet your requirements, check the following items before fine-tuning:

- The input design
Ensure that the input design meets the requirements for clock tree synthesis.
- TLUPlus models
- The clock root
Verify that you have correctly defined the clock root, including setting a driving cell if the clock root is a top-level input port. Verify that the attributes of the clock root, such as the input transition time and output capacitance, are correct. Verify that the library model for the root cells is accurate.

- The clock sinks

Run the `report_clock_tree -exceptions` command to verify that the clock tree exceptions (both implicit and explicit) are correctly set.

- The design rule constraints

To verify the design rule constraints and check that they are realistic, use the `report_clock_tree` command to report on the clock tree (or choose Clock > Report Clock Tree in the GUI). If your clock trees have too many cells or levels, this typically indicates that the design rule constraints are too tight.

Resynthesizing the Clock Trees

You can resynthesize the clock trees either by using the `clock_opt` command (as described in “[Implementing Clock Trees](#)” on page 5-84) or by using standalone clock tree synthesis (as described in “[Performing Clock Tree Synthesis](#)” on page 5-98).

Modifying the Nondefault Routing Rule

If your design is congested, you can modify the nondefault routing rule of an existing clock tree. To modify the nondefault routing rule, use the `mark_clock_tree -routing_rule` command (or choose Clock > Mark Clock Tree in the GUI and specify the routing rule in the “Routing rule” box). In the same way that you specify the nondefault rule for unsynthesized clock trees, you can use the default routing rule for the nets connected to the clock sinks by specifying the `-use_default_routing_for_sinks level_count` option (or by selecting “Routing rule” in the GUI and specifying the level count in the “Use default routing for sinks at level” box). In addition, you can specify the layers to use for clock routing by specifying the `-layer_list layers` option (or by selecting the layers in the GUI). For more information about the clock tree routing options, see “[Setting Clock Tree Routing Options](#)” on page 5-54.

Note:

If you specify an undefined routing rule or layer, the `mark_clock_tree` command exits with an error message and does not change the existing settings.

When you change the nondefault routing rule, the tool updates the RC data by running the `extract_rc` command to perform extraction using the new routing rule information.

Modifying Clock Trees in the GUI

You can modify clock trees in the GUI by using the interactive clock tree synthesis capability. You can use interactive clock tree synthesis for both preroute and postroute designs.

Note:

Interactive clock tree synthesis considers a design to be postroute only if all clock nets are routed.

Interactive clock tree synthesis allows you to do what-if analysis on the clock trees and then commit the changes, if they meet your requirements.

You can invoke the interactive clock tree synthesis capabilities from the layout menu (by choosing Clock > ECO) or by selecting an object in the interactive clock tree synthesis window and right-clicking to open the pop-up menu. [Table 5-19](#) shows the capabilities supported by interactive clock tree synthesis and the menu option used to invoke this capability.

Table 5-19 Interactive Clock Tree Synthesis Capabilities

Task	Supported phases	Menu option
Insert buffer	preroute	CTS Insert Buffer
Remove buffer	preroute	CTS Remove Buffer
Move buffer or gate	preroute	CTS Move Cell
Move a clock tree branch to another point in the same clock tree	preroute	CTS Re-parent
Size buffer or gate	preroute or postroute	CTS Size Cell

When you make changes, you can update the timing by clicking Update Timing in the associated dialog box. If the change meets your requirements, click Legalize to legalize the placement and commit the changes. If the change does not meet your requirements, click Undo to remove the changes.

If you are using interactive clock tree synthesis on a postroute design, you can also perform ECO routing (by clicking Route ECO) and extraction (by clicking Extract RC) after committing the sizing changes.

Note:

If new nets are added to the design as a result of the modifications, the tool does not propagate nondefault routing rules to these new nets. You can use the `mark_clock_tree` command to reapply the nondefault routing rule to the clock tree, including the new nets.

Inserting a Buffer

To insert a buffer using interactive clock tree synthesis,

1. Select CTS Insert Buffer.

The CTS Insert Buffer dialog box appears.

2. Specify the following:

- Whether you want to insert a buffer or an inverter pair
- To which clock pin you want to insert the buffer

If you specify an output pin, the new buffer drives all fanouts of the specified pin. If you specify an input pin, the new buffer drives only that pin.

- The library reference cell to use
If you specify a pattern, the tool locates all library reference cells matching the specified pattern.
- (Optional) The names for the new cells and nets
- The physical location of the inserted buffer

Note:

You can also specify the physical location of the inserted buffer by left-clicking in the layout window

- (Optional) the colors to use to highlight the inserted buffer in the layout window

3. Click “Insert Buffer” to insert the buffer.
4. Click “Update Timing” to perform what-if analysis for this change.
5. Click Legalize to commit this change or click Undo to undo this change.

Removing a Buffer

To remove a buffer using interactive clock tree synthesis,

1. Select CTS Remove Buffer.

The CTS Remove Buffer dialog box appears.

2. Specify the buffer to remove.
3. Click “Remove Buffer” to remove the buffer.
4. Click “Update Timing” to perform what-if analysis for this change.
5. Click Legalize to commit this change or click Undo to undo this change.

Moving a Buffer or Gate

To move a buffer or gate using interactive clock tree synthesis,

1. Select CTS Move Cell.

The CTS Move Cell dialog box appears.

2. Click “Move Cell” to enter move mode in the layout window.
3. Move the cell (a clock buffer or clock gate) in the layout window.
4. Click Legalize to commit this change or click Undo to undo this change.

Reparenting a Subtree

To reparent a subtree using interactive clock tree synthesis,

1. Select CTS Re-parent.

The CTS Re-parent dialog box appears.

2. Specify the following:
 - The subtree root pin
 - The new driving pin for the subtree
3. Click Re-parent to perform what-if analysis for this change.
4. Click Legalize to commit this change or click Undo to undo this change.

The new driving pin that you specify for the reparented subtree must meet the following requirements:

- It must be in the same clock tree as the existing driving pin.
- It must be in the same logical hierarchy as the existing driving pin.
- It must not be a don’t touch subtree pin.
- It must not drive a don’t buffer net.

Sizing a Buffer or Gate

To size a buffer or gate using interactive clock tree synthesis,

1. Select CTS Size Cell.

The CTS Size Cell dialog box appears.

2. Specify the following:

- The buffer or gate to be sized (the cell must be on the clock path)
- The library reference cell that can be used

3. Click “Size Cell” to perform what-if analysis for this change.

4. Click Legalize to commit this change or click Undo to undo this change.

Analyzing and Refining the Design

After you are satisfied with the clock tree synthesis results, analyze the QoR of the entire design by reporting on constraints (use the `report_constraint` command or choose Design > Report Constraints in the GUI) and timing (use the `report_timing` command).

Use the reports to check the following parameters:

- Worst negative slack (WNS)
- Total negative slack (TNS)
- Design rule constraint violations

6

Routing Using Zroute

This chapter describes the routing capabilities of Zroute, which is the default router for all IC Compiler packages, with the exception of the ICC-XP package. Zroute is architected for multicore hardware and efficiently handles advanced design rules for 45 nm and below technologies and design-for-manufacturing (DFM) tasks. For information about the classic router, see the *IC Compiler Classic Router User Guide*.

Note:

The IC Compiler-PC package does not include signal routing capabilities. If you are using the IC Compiler-PC package, you need to export the design so that you can perform signal routing with another tool. (See “[Saving a Design](#)” on page 3-60.)

This chapter contains the following sections:

- [Zroute Features](#)
- [Basic Zroute Flow](#)
- [Prerequisites for Routing](#)
- [Creating Design-Specific Via Masters](#)
- [Checking Routability](#)
- [Setting Routing Constraints](#)
- [Setting Routing Options](#)
- [Routing Clock Nets](#)

- Routing Critical Nets
- Routing Secondary Power and Ground Pins
- Routing Signal Nets
- Performing Focal Optimization
- Performing Final Stage Leakage-Power Recovery
- Performing ECO Routing
- Cleaning Up Routed Nets
- Saving the Routing Information
- Analyzing the Routing Results
- Routing Nets and Buses in the GUI
- Postroute RC Extraction

Zroute Features

Zroute has five routing engines: global routing, track assignment, detail routing, ECO routing, and routing verification. You can invoke global routing, track assignment, and detail routing by using the `route_opt` core command; by using task-specific commands; or by using an automatic routing command. You invoke ECO routing and routing verification by using task-specific commands.

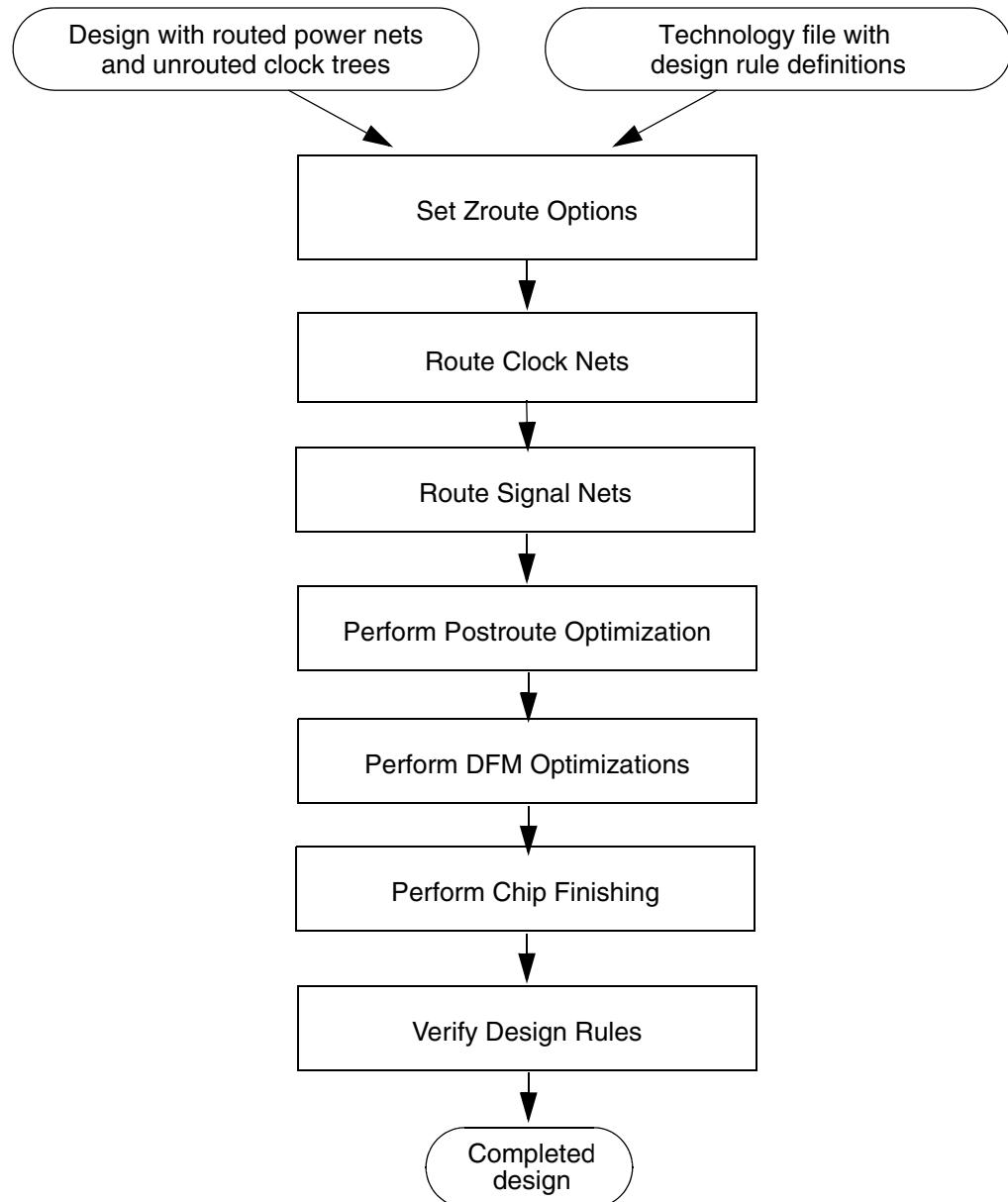
Zroute includes the following main features:

- Multithreading on multicore hardware for all routing steps, including global routing, track assignment, and detail routing
- A realistic connectivity model where Zroute recognizes electrical connectivity if the rectangles touch; it does not require the center lines of wires to connect
- A dynamic maze grid that permits Zroute to go off-grid to connect pins, while retaining the speed advantages of gridded routers
- A polygon manager, which allows Zroute to recognize polygons and to understand that design rule checks (DRCs) are aimed at polygons
- Concurrent optimization of design rules, antenna rules, wire optimization, and via optimization during detail routing
- Concurrent redundant via insertion during detail routing
- Support for soft rules built into global routing, track assignment, and detail routing
- Timing- and crosstalk-driven global routing, track assignment, detail routing, and ECO routing
- Intelligent design rule handling, including merging of redundant design rule violations and intelligent convergence
- Net group routing with layer constraints and nondefault routing rules
- Clock routing
- Route verification
- Optimization for DFM and design-for-yield (DFY) using a soft rule approach

Basic Zroute Flow

[Figure 6-1](#) shows the basic Zroute flow, which includes clock routing, signal routing, DFM optimizations, and route verification.

Figure 6-1 Basic Zroute Flow



Prerequisites for Routing

Before you can run Zroute, you must ensure that the design and physical library meet the following requirements:

- Library requirements

Zroute gets all of the design rule information from the Milkyway technology file; therefore, you must ensure that all design rules are defined in the technology file before you start routing.

In addition, Zroute uses only default vias when routing nets, except when nondefault vias are explicitly specified in a fat via table in the technology file, a nondefault routing rule defined by the `define_routing_rule` command, or a via mapping table defined by the `define_zrt_redundant_vias` command. Make sure that all via masters that you want Zroute to use are defined as default vias (`isDefaultContact` attribute is 1) in the technology file. If you need to use a via master that is not defined in the technology file, you can create a design-specific via master by using the `create_via_master` command, as described in [“Creating Design-Specific Via Masters” on page 6-6](#).

For more information about the technology file and defining routing design rules, see the *IC Compiler Technology File and Routing Rules Reference Manual*.

- Design requirements

Before you 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 [Chapter 5, “Clock Tree Synthesis.”](#)
- 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 [“Checking Routability”](#) in the following section.

Note:

You can use the prerouter to preroute signal nets before running Zroute. The prerouted signal nets are marked as user nets. By default, Zroute does not rip up and reroute these user nets. Zroute makes only minor changes to correct DRC violations.

Creating Design-Specific Via Masters

If you need to use a via master that is not defined in the technology file, you can create a design-specific via master by using the `create_via_master` command. After it is created, the via master can be used anywhere in the design, similar to a `ContactCode` definition in the technology file. The via master is stored as an object together with the design cell in the Milkyway database, so it can be used only in that design.

The `create_via_master` command can create both parameter-based and rectangle-based via masters.

- Parameter-based via masters can be used for the following purposes:
 - Clock or signal routing using nondefault routing rules
To define parameter-based design-specific via masters as nondefault vias, use the `-via_cuts` option with the `define_routing_rule` command, as described in [“Defining Nondefault Vias” on page 6-21](#).
 - Redundant via insertion
To define the via masters used for redundant via insertion, including parameter-based design-specific via masters, use the `define_zrt_redundant_vias` command, as described in [“Defining a Customized Via Mapping Table” on page 7-27](#).
 - Power and ground routing with advanced via rules
To define the parameter-based design-specific via masters used for power and ground routing, use the `set_preroute_advanced_via_rule` command.
- Rectangle-based via masters must be H-shaped vias and can be used only for redundant via insertion
To define the via masters used for redundant via insertion, including rectangle-based design-specific via masters, use the `define_zrt_redundant_vias` command, as described in [“Defining a Customized Via Mapping Table” on page 7-27](#).

The following sections describe how to create these types of via masters.

Creating Parameter-Based Via Masters

To create a parameter-based design-specific via master, use the following syntax:

```
create_via_master
  -name string
  -cut_layer_name layer
  -lower_layer_name layer
  -upper_layer_name layer
  -cut_width float
  -cut_height float
  -lower_layer_enc_width float
  -lower_layer_enc_height float
  -upper_layer_enc_width float
  -upper_layer_enc_height float
  -min_cut_spacing float
  [-quiet]
```

For example,

```
icc_shell> create_via_master -name design_via1_HV -cut_layer_name VIA12 \
  -cut_height 0.05 -cut_width 0.05 \
  -lower_layer_name METAL1 \
  -lower_layer_enc_width 0.02 -lower_layer_enc_height 0.0 \
  -upper_layer_name METAL2 \
  -upper_layer_enc_width 0.0 -upper_layer_enc_height 0.02 \
  -min_cut_spacing 0.05
```

To report information about the parameter-based via masters, use the `report_via_master` command.

Creating Rectangle-Based Via Masters

To create a rectangle-based design-specific via master, use the following syntax:

```
create_via_master
  -name string
  -cut_layer_name layer
  -lower_layer_name layer
  -upper_layer_name layer
  -rectangles {{layer{rectangle}}...}
  [-quiet]
```

You can specify multiple rectangles per layer; however, you cannot specify more than two metal layers.

For example, to create a design-specific rectangle-based H-shape via master, use a command similar to the following:

```
icc_shell> create_via_master -name design_H_shape_via \
    -cut_layer_name VIA12 \
    -lower_layer_name METAL1 -upper_layer_name METAL2 \
    -rectangles { {VIA12 {0.035 -0.035 0.100 0.035}}
    {VIA12 {-0.100 -0.035 -0.035 0.035}}
    {METAL1 {-0.130 -0.035 0.130 0.035}}
    {METAL2 {-0.100 -0.065 -0.035 0.065}}
    {METAL2 {0.035 -0.065 0.100 0.065}}
    {METAL2 {-0.100 -0.035 0.100 0.035}}
}
```

To report information about the rectangle-based via masters, use the `report_via_master` command.

Checking Routability

After placement is completed, you can use the `check_zrt_routability` command (or choose Route > Check Routability in the GUI) to check whether your design is ready for detail routing.

By default, this command checks for

- Blocked standard cell ports

A standard cell port is considered blocked if none of its physical pins is accessible.

A standard cell pin is considered accessible if the pin contains a via that extends to a neighboring layer, there is a path on the pin layer that is at least as long as the search range distance, or there is a shorter path on the pin layer that ends at a via to a neighboring layer. By default, the search range distance is two times the layer pitch. To change this distance, use the `-standard_cell_search_range` option to specify a different pitch multiplier, up to a maximum of 10. If you specify a value larger than 10, the command sets the value to 10.

By default, the tool does not check whether via connections are fully inside the standard cell pins. To enable this check, use the `-connect_standard_cells_within_pins` true option.

To disable the checking of blocked standard cell ports, use the `-check_standard_cell_blocked_ports false` option.

- Blocked top-level or macro cell ports

A top-level or macro cell port is considered blocked if none of its physical pins is accessible.

A top-level or macro pin is considered accessible if a legal path can be extended from the pin to a certain distance around it. This path can be just on the pin layer or can extend to a neighboring layer by a single via. By default, the distance is 10 times the layer pitch. To change this distance for same-layer paths, use the `-blocked_range` option to specify a different pitch multiplier, up to a maximum of 40. To change this distance for neighboring-layer paths through a via, use the `-blocked_range_via_side` option to specify a different pitch multiplier, up to a maximum of 40. If you specify a value larger than 40 for either of these options, the command sets its value to 40.

By default, the command checks whether a pin is accessible in either the horizontal or vertical direction. To require that the pin is accessible in the preferred direction for its layer, use the `-obey_direction_preference true` option.

To disable the checking of blocked top-level or macro cell ports, use the `-check_non_standard_cell_blocked_ports false` option.

- Out-of-boundary pins

This check verifies that all pins are within the design boundary.

To disable the checking of out-of-boundary pins, use the `-check_out_of_boundary false` option.

- Minimum grid violations

This check verifies that all pins, including those within library cells, are on the minimum grid, as defined by the `gridResolution` attribute in the technology file.

To disable the checking of minimum grid violations, use the `-check_min_grid false` option.

The `check_zrt_routability` command also supports the following optional checks:

- Blocked power or ground ports

To enable this check, use the `-check_pg_blocked_ports true` option.

- Blocked ports on frozen nets

To enable this check, use the `-check_frozen_net_blocked_ports true` option.

- Blocked unconnected pins

To enable this check, use the `-check_no_net_pins true` option.

The `check_zrt_routability` command supports the following additional pin connection controls that apply to all pin access checks:

- Pin access edges

During blockage, pin, and via extraction (BPV), the FRAM views are annotated with information about the pin access edges. By default, the `check_zrt_routability` command ignores the pin access edges and allows pin connections at any point. You can restrict pin connections to the defined access edges by setting the `-obey_access_edges` option to `true`. You can further restrict pin connections to the access edge mark, which can be a narrow rectangle, a short line, or even a single point, by setting the `-access_edge_whole_side` option to `true`. In addition, the command can report unconnected pins that do not have an access edge defined. To enable this check, use the `-report_no_access_edge true` option.

- Via rotation

By default, pin connections can use rotated vias. To disallow pin connections that use rotated vias, set the `-allow_via_rotation` option to `false`.

By default, this command considers the following routing layer constraints when checking for blocked ports:

- The global minimum and maximum routing layer constraints set by the `-min_routing_layer` and `-max_routing_layer` options of the `set_ignored_layers` command
- The net-specific minimum and maximum routing layer constraints set by the `-min_layer_name` and `-max_layer_name` options of the `set_net_routing_layer_constraints` command
- The freeze layer constraints set by the `-freeze_layer_by_layer_name` and `-freeze_via_to_frozen_layer_by_layer_name` options of the `set_route_zrt_common_options` command

To ignore the routing layer constraints during the blocked port checks, use the `-honor_layer_constraints false` option.

When you run the `check_zrt_routability` command, it saves the reported violations in an error view. By default, the error view is named after the top-level design (*top_design_name.err*). To specify the name of the error view, use the `-error_view` option. To report the location of each error, use the `report_error_coordinates` command. 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-94 and the “Examining Routing and Verification Errors” topic in IC Compiler Help.

Setting Routing Constraints

Routing constraints provide guidance during routing. [Table 6-1](#) describes the types of guidance provided by the routing constraints.

Table 6-1 Routing Constraints

Guidance	Description
Controlling the layers used for routing	For information about specifying the routing layers, see “ Specifying the Routing Layers .”
Setting stricter minimum width and spacing rules	To set stricter minimum width and spacing rules, define a nondefault routing rule and apply it to the affected nets. For information about defining and applying nondefault routing rules, see “ Using Nondefault Routing Rules .”
Controlling the routing direction	<p>You can specify the preferred routing direction for specific layers or for specific regions.</p> <ul style="list-style-type: none"> For information about specifying the preferred routing direction for specific layers, see “Setting the Preferred Routing Direction.” To control the routing direction in a specific region, use either a preferred-direction-only route guide or a switch-preferred-direction route guide. For information about route guides, see “Defining Route Guides.”
Limiting the number of edges in the nonpreferred routing direction	To limit the number of edges in the nonpreferred routing direction, use a maximum-number-of-pattern route guide. For information about route guides, see “ Defining Route Guides .”
Controlling off-grid routing	<p>By default, off-grid routing is allowed for wires or vias unless the <code>onWireTrack</code> or <code>onGrid</code> attribute is set to 1 for the layer in the technology file.</p> <p>You can prevent or discourage off-grid routing, as described in “Controlling Off-Grid Routing” on page 6-27.</p>
Preventing routing in specific regions	<p>To prevent routing in a specific region, define either a route guide or a routing blockage. Route guides are defined on metal or via layers and are the preferred method. Routing blockages are defined on blockage layers and require extra settings to be recognized by the router.</p> <ul style="list-style-type: none"> For information about route guides, see “Defining Route Guides.” For information about routing blockages, see “Defining Routing Blockages.”

Table 6-1 Routing Constraints (Continued)

Guidance	Description
Restricting routing to specific regions	To restrict routing to a specific region, use a routing corridor. For information about routing corridors, see “ Defining Routing Corridors .”
Prioritizing routing regions	To prioritize routing regions, use an access preference route guide. For information about route guides, see “ Defining Route Guides .”
Controlling the routing density	To control the routing density for specific layers, use a utilization route guide. For information about route guides, see “ Defining Route Guides .”
Encouraging river routing	To encourage river routing on specific layers, use a single-layer route guide. For information about route guides, see “ Defining Route Guides .”
Controlling the pin connections	<p>You can control both the allowed types of pin connections and the pin tapering.</p> <ul style="list-style-type: none"> • For information about controlling the allowed types of pin connections, see “Controlling Pin Connections.” • For information about controlling pin tapering, see “Controlling Pin Tapering.”
Restricting the extent of rerouting	For information about restricting the extent of rerouting for specific nets, see “ Setting the Rerouting Mode .”
Controlling the partitions used for DRC fixing	To control whether violations are fixed as a single switch box, use a route guide. For information about route guides, see “ Defining Route Guides .”

Specifying the Routing Layers

The IC Compiler tool lets you specify which layers can be ignored for routing (global layer constraints) 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-specific routing layer constraints).

Specifying the Global Layer Constraints

To specify the global layer constraints, 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
```

By default, when you set global routing layer constraints, they are soft constraints.

- To change the minimum routing layer constraint to a hard constraint or to allow the use of lower layers only for pin connections, use the `set_route_zrt_common_options` command to set the `-global_min_layer_mode` common route option to `hard` or `allow_pin_connection`, respectively.
- To change the maximum routing layer constraint to a hard constraint or to allow the use of higher layers only for pin connections, use the `set_route_zrt_common_options` command to set the `-global_max_layer_mode` common route option to `hard` or `allow_pin_connection`, respectively.

To use fewer layers for RC estimation and congestion analysis than those used for routing, use the `-rc_congestion_ignored_layers` option to specify the layers to be ignored for RC estimation and congestion analysis. For example, to use layers M2 through M7 for routing and layers M3 through M7 for RC estimation and congestion analysis, use the following command:

```
icc_shell> set_ignored_layers \
    -min_routing_layer M2 -max_routing_layer M7 \
    -rc_congestion_ignored_layers {M1 M2 M8 M9}
```

To use more layers for RC estimation and congestion analysis than for routing, use the `remove_ignored_layers` command to remove the layer restriction for RC estimation and

congestion analysis. For example, to use layers M2 through M7 for routing and layers M2 through M8 for RC estimation and congestion analysis, use the following commands:

```
icc_shell> set_ignored_layers \
    -min_routing_layer M2 -max_routing_layer M7 \
icc_shell> remove_ignored_layers M8
```

Reporting Global Layer Constraints

To report the ignored layers, use the `report_ignored_layers` command.

Modifying the Global Layer Constraints

To modify the global layer constraints, you must first remove all of the global layer constraints by using the `remove_ignored_layers -all` command, and then define the new constraints by using the `set_ignored_layers` command, as shown in the following example:

```
icc_shell> report_ignored_layers
Ignored layers in congestion analysis and RC estimation: M1 M8 M9
Min_routing_layer: M2
Max_routing_layer: M7

icc_shell> remove_ignored_layers -all
icc_shell> set_ignored_layers -max_routing_layer M9 -min_routing_layer M2
icc_shell> report_ignored_layers
Ignored layers in congestion analysis and RC estimation: M1
Min_routing_layer: M2
Max_routing_layer: M9
```

Removing the Global Layer Constraints

To remove global layer constraints, use the `remove_ignored_layers` command. By default, the command removes the ignored setting from the specified routing layers (or all routing layers if you use the `-all` option), so that they are used for congestion analysis and RC estimation.

To remove the global minimum layer constraint, use the `-min_routing_layer` option with the `remove_ignored_layers` command. To remove the global maximum layer constraint, use the `-max_routing_layer` option with the `remove_ignored_layers` command.

Specifying Net-Specific Routing Layer Constraints

To specify net-specific routing layer constraints, use the `set_net_routing_layer_constraints` command (or choose Route > Routing Setup > Set Net Routing Layer Constraints in the GUI).

To specify the minimum and maximum routing layers for specific nets, use the `-min_layer_name` and `-max_layer_name` options, respectively. For example, to use layers M2 through M7 for routing net n1, use the following command:

```
icc_shell> set_net_routing_layer_constraints [get_nets n1] \
    -min_layer_name M2 -max_layer_name M7
```

By default, all net-specific maximum routing layer constraints are treated as hard constraints and all net-specific minimum routing layer constraints are treated as soft constraints. You can override the default strength for these constraints, as well as the strength for specific nets.

- To set the default strength for the net-specific maximum routing layer constraints, use the `set_route_zrt_common_options` command to set the `-net_max_layer_mode` common route option.
- To set the default strength for the net-specific minimum routing layer constraints, use the `set_route_zrt_common_options` command to set the `-net_min_layer_mode` common route option.
- To set the net-specific strength for the maximum routing layer constraint, use the `-max_layer_mode` option when you define the constraint with the `set_net_routing_layer_constraints` command.
- To set the net-specific strength for the minimum routing layer constraints, use the `-min_layer_mode` option when you define the constraint with the `set_net_routing_layer_constraints` command.

Set these options to `hard` to specify a hard constraint, `soft` to specify a soft constraint, or `allow_pin_connection` to allow the use of lower layers only for pin connections.

For soft net-specific routing layer constraints, you can control the relative cost for violating the constraint. The default cost is `medium`. To increase the relative cost and have the tool work harder to avoid violations, set the cost to `high`. To decrease the relative cost and possibly reduce the runtime but have more violations, set the cost to `low`.

- For soft net-specific maximum routing layer constraints,
 - Set the default cost by using the `set_route_zrt_common_options` command to set the `-net_max_layer_mode_soft_cost` common route option.
 - Set the net-specific cost by setting the `-max_layer_mode` option when you define the constraint with the `set_net_routing_layer_constraints` command.

- For soft net-specific minimum routing layer constraints,
 - Set the default cost by using the `set_route_zrt_common_options` command to set the `-net_min_layer_mode_soft_cost` common route option.
 - Set the net-specific cost by setting the `-min_layer_mode` option when you define the constraint with the `set_net_routing_layer_constraints` command.

Reporting Net-Specific Routing Layer Constraints

To report the routing layer constraints for specific nets, use the `report_net_routing_layer_constraints` command.

```
icc_shell> report_net_routing_layer_constraints [get_nets *]
```

To output a Tcl script that contains the `set_net_routing_layer_constraints` commands that are used to define the routing layer constraints for the specified nets, use the `-output` option when you run the `report_net_routing_layer_constraints` command.

Removing Net-Specific Routing Layer Constraints

To remove net-specific routing layer constraints, use the `remove_net_routing_layer_constraints` command.

Using Nondefault Routing Rules

Zroute supports the use of nondefault routing rules, both for routing and for shielding.

- For routing, you can use nondefault routing rules to define stricter wire width and spacing rules, to define the tapering distance, and to specify the vias used when routing nets with nondefault routing rules.
- For shielding, you can use nondefault routing rules to define the minimum width and spacing rules.

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). You can assign multiple nondefault routing rules to a net.

When you define a nondefault routing rule, you must specify a name for the nondefault routing rule. If you rerun the `define_routing_rule` command using a previously defined nondefault routing rule, the new definition overwrites the existing definition. To change the

definition for an existing rule, use the `remove_routing_rules` command to remove the rule and then use the `define_routing_rule` command to redefine the rule.

The following sections describe the tasks associated with defining nondefault routing rules.

Defining Minimum Wire Width Rules

You can use a nondefault routing rule to define minimum wire width rules that are stricter than the minimum width rules defined in the Milkyway technology file. Nondefault minimum width rules are hard constraints, which must be met during routing.

Note:

The minimum width defined in a nondefault routing rule applies to all metal segments, including via enclosures. To avoid DRC violations, ensure that the enclosures for nondefault vias meet the minimum width rule.

The syntax for specifying nondefault wire width rules is

```
define_routing_rule rule_name
    [-default_reference_rule | -reference_rule_name ref_rule]
    -widths {layer1 width1 layer2 width2 ... layern widthn}
```

You can define a single width value per layer. The minimum wire width for any layers not specified in the `-widths` option is determined from the reference rule, either the default routing rule or the reference rule specified in the `-reference_rule_name` option.

For example, to define a nondefault routing rule named `new_width_rule` that uses the default routing rule as the reference rule and defines nondefault width rules for metal1 and metal4, use the following command:

```
icc_shell> define_routing_rule new_width_rule -widths {m1 0.8 m4 0.9}
```

Defining Wire Spacing Rules

You can use a nondefault routing rule to define minimum wire spacing rules that are stricter than the rules defined in the Milkyway technology file. Nondefault wire spacing rules can be defined as hard constraints, which must be met, or as soft constraints, which Zroute tries to meet. If you define more than one spacing rule per layer, you must assign a weight level to each nondefault spacing rule that defines the strength of the rule and assign a routing effort to each soft weight level. You can also define a length threshold for the nondefault spacing rules to ignore violations of the nondefault spacing rules for short parallel distances.

Note:

The spacing rules defined in the Milkyway technology file are always considered hard constraints, whether specified in the `define_routing_rule` command or not.

The syntax for specifying nondefault wire spacing rules, their weight levels, and their length thresholds is

```
define_routing_rule rule_name
  -spacings { layer1 {spacing11 spacing12 ... spacing1n}
               layer2 {spacing21 spacing22 ... spacing2n}
               ...
               layern {spacingn1 spacingn2 ... spacingnn} }
  -spacing_weight_levels { layer1 {weight11 weight12 ... weight1n}
                            layer2 {weight21 weight22 ... weight2n}
                            ...
                            layern {weightn1 weightn2 ... weightnn} }
  -spacing_length_thresholds { layer1 {length11 length12 ... length1n}
                               layer2 {length21 length22 ... length2n}
                               ...
                               layern {lengthn1 lengthn2 ... lengthnn} }
```

The minimum wire spacing for any layers not specified in the `-spacings` option is determined from the reference rule, either the default routing rule or the reference rule specified in the `-reference_rule_name` option.

You can define multiple spacing values per layer. If you specify more than one spacing value per layer, you must assign a weight to each spacing value by using the `-spacing_weight_levels` option. The valid weight values are `low`, `medium`, `high`, and `hard`.

For example, to define a nondefault routing rule named `new_spacing_rule` that uses the default routing rule as the reference rule and defines nondefault spacing rules for metal1 and metal4, use the following command:

```
icc_shell> define_routing_rule new_spacing_rule \
  -spacings { m1 {0.12 0.24} m4 {0.14 0.28} } \
  -spacing_weight_levels { m1 {hard medium} m4 {hard medium} }
```

When you use soft weight levels to assign relative strengths to the soft spacing rules, you must assign a routing effort for each soft weight level by using the `set_route_zrt_common_options` command to set the `-soft_rule_weight_to_effort_level_map` common route option.

The syntax for specifying the routing effort for the soft weight levels is

```
set_route_zrt_common_options
  -soft_rule_weight_to_effort_level_map
    { {weight effort} ... }
```

Each `weight` argument can be one of `low`, `medium`, or `high`. You should specify each weight level only one time. Each `effort` argument can be one of `off`, `low`, `medium`, or `high`. If you specify a weight level multiple times, the router uses the last specification.

For each weight level that you specify, use one of the following keyword values to specify its effort level:

- `off` (the default)

Zroute does not perform rip-up and reroute passes to resolve soft spacing rule violations.

- `low`

Zroute uses a small number of rip-up and reroute passes to resolve soft spacing rule violations.

- `medium`

Zroute uses a medium number of rip-up and reroute passes to resolve soft spacing rule violations. Note that you cannot specify this effort level for the `low` weight level.

- `high`

Zroute treats soft spacing rule violations the same as regular design rule violations during rip up and reroute. Note that you cannot specify this effort level for the `low` weight level.

If you do not specify an effort level for a weight level, the tool uses the default effort level of `off` and does not fix spacing rule violations with that weight level.

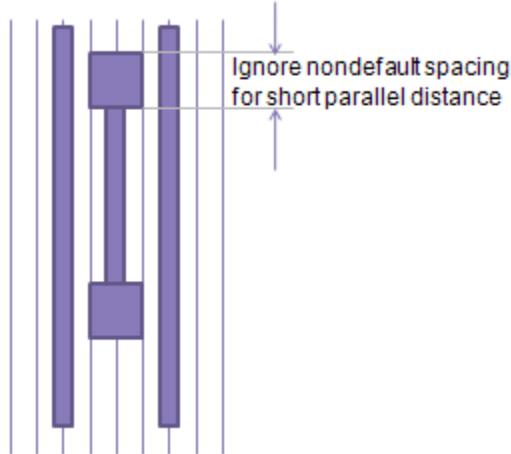
For example, to assign low routing effort to low-weight soft spacing rules, medium routing effort to medium-weight soft spacing rules, and high routing effort to high-weight soft spacing, use the following command:

```
icc_shell> set_route_zrt_common_options \
    -soft_rule_weight_to_effort_level_map \
    { {low low} {medium medium} {high high} }
```

By default, nondefault spacing rules are honored between signal wires and other signal wires, PG wires, and blockages, but not between signal wires and shield wires for PG nets or between shapes of the same signal wire.

- To increase the flexibility of DRC convergence without adversely affecting crosstalk, you can ignore violations of the nondefault spacing rules for short parallel distances, as shown in [Figure 6-2](#).

Figure 6-2 Ignoring Nondefault Spacing Rule Violations



You define the length thresholds within which to ignore the nondefault spacing violations by using the `-spacing_length_thresholds` option. The length threshold values are in microns and must have a one-to-one correspondence with the spacing entries specified in the `-spacings` option.

For example,

```
icc_shell> define_routing_rule new_rule \
    -spacings { m1 {0.09 0.15 0.2}
                 m2 {0.09 0.15 0.2}
                 m3 {0.09 0.15 0.2} } \
    -spacing_weight_levels { m1 {hard medium low}
                             m2 {hard medium low}
                             m3 {hard medium low} } \
    -spacing_length_thresholds { m1 {0.01 0 0}
                                 m2 {0.01 0 0}
                                 m3 {0.01 0 0} }
```

- To ignore the nondefault spacing rules between signal wires and PG wires, use the `set_route_zrt_common_options` command to set the `-ignore_var_spacing_to_pg` common route option to `true`.
- To ignore the nondefault spacing rules between signal wires and blockages, use the `set_route_zrt_common_options` command to set the `-ignore_var_spacing_to_blockage` common route option to `true`.

- To honor the nondefault spacing rules between signal wires and shield wires for PG nets, use the `set_route_zrt_common_options` command to set the `-ignore_var_spacing_to_shield` common route option to `false`.
- To honor the nondefault spacing rules between shapes of the same signal wire, use the `set_route_zrt_detail_options` command to set the `-var_spacing_to_same_net` detail route option to `true`.

Defining Via Spacing Rules

You can use a nondefault routing rule to define minimum via spacing rules that are stricter than the rules defined in the Milkyway technology file. Nondefault via spacing rules are hard constraints that must be met.

Note:

The spacing rules defined in the Milkyway technology file are always considered hard constraints whether specified in the `define_routing_rule` command or not.

The syntax for specifying nondefault via spacing rules is

```
define_routing_rule rule_name
  -via_spacings { {layer1 layer2 spacing} ... }
```

The minimum spacing between vias on any layer combinations not specified in the `-via_spacings` option is determined from the Milkyway technology file. For vias on the same layer, the router uses the minimum spacing defined in the `Layer` section for the via layer. For vias on different layers, the router uses the minimum spacing defined in the `DesignRule` section for the via layer combination.

For example, to define a nondefault routing rule named `via_spacing_rule` that defines nondefault spacing rules between vias on the V1 layer and between vias on the V1 and V2 layers, and uses the minimum spacing rules defined in the technology file between vias on all other layer combinations, use the following command:

```
icc_shell> define_routing_rule via_spacing_rule \
  -via_spacings {{V1 V1 2.3} {V1 V2 3.2}}
```

Defining Nondefault Vias

You use a nondefault routing rule to define the vias to use when routing nets with nondefault routing rules. You can define the nondefault vias either by using the `-cuts` option or by using the `-via_cuts` option. When you use the `-cuts` options, the tool determines the suitable vias from the technology file based on the specification in the `-cuts` option and the rules defined in the technology file. When you use the `-via_cuts` option, you explicitly specify the nondefault vias, including the allowed cut numbers and rotation for each via. The vias can

be via masters defined in the technology file or parameter-based design-specific via masters created by the `create_via_master` command. For information about creating design-specific via masters, see “[Creating Design-Specific Via Masters](#)” on page 6-6.

The syntax for specifying nondefault vias using the `-cuts` option is

```
define_routing_rule rule_name
  -cuts { {cut_layer1 {cut_name1, ncuts} {cut_name2, ncuts} ...}
          {cut_layer2 {cut_name1, ncuts} {cut_name2, ncuts} ...}
          ...
          {cut_layerN {cut_name1, ncuts} {cut_name2, ncuts} ...}
      }
```

The `cut_layer` arguments refer to the via layer names in the technology file and the `cut_name` arguments refer to the cut names defined in the `cutNameTbl` attribute in the associated `Layer` section in the technology file. You can specify multiple cut names per layer. For each cut name, you must specify the minimum number of cuts, which must be an integer between 1 and 255.

The tool searches for the vias defined in the `ContactCode` section of the technology file that meet the rules defined in the technology file for the specified cut name, such as the `cutWidthTbl`, `cutHeightTbl`, and `fatTblFatContactNumber`. If the fat metal contact rule is not defined for a via layer, the tool searches for the default vias that meet the cut width and height requirements. The minimum number of cuts required is the larger of the `ncuts` value in the `-cuts` option and the value defined in the `fatTblFatContactMinCuts` attribute. For the selected vias, the tool always allows both the rotated and the unrotated orientation for the via.

For example, assume the following information is defined in the technology file:

```
Layer "VIA1" {
    fatTblThreshold      = ( 0, 0.181, 0.411 )
    fatTblFatContactNumber = ( "2,3,4,5,6 ", "5,6,20", "5,6,20" )
    fatTblFatContactMinCuts = ( "1,1,1,1,1", "1,1,1", "2,2,2" )

    cutNameTbl     = ( Vsq, Vrect )
    cutWidthTbl    = ( 0.05, 0.05 )
    cutHeightTbl   = ( 0.05, 0.13 )
    ...
}

ContactCode "VIA12_LH" {
    contactCodeNumber = 5
    cutWidth         = 0.13
    cutHeight        = 0.05
    ...
}
```

```

ContactCode "VIA12_LV" {
    contactCodeNumber = 6
    cutWidth          = 0.05
    cutHeight         = 0.13
    ...
}
ContactCode "VIA12_P" {
    contactCodeNumber = 20
    cutWidth          = 0.05
    cutHeight         = 0.05
    ...
}

```

If you use the following command,

```
icc_shell> define_routing_rule cut_rule -widths { M1 0.2 M2 0.25 } \
    -cuts {VIA1 {Vrect 1}}
```

The tool selects the following vias: {VIA12_LH 1x1 R}, {VIA12_LH 1x1 NR}, {VIA12_LV 1x1 R}, {VIA12_LV 1x1 NR}, {VIA12_LH 1x2 R}, {VIA12_LH 1x2 NR}, {VIA12_LH 2x1 R}, {VIA12_LH 2x1 NR}, {VIA12_LV 1x2 R}, {VIA12_LV 1x2 NR}, {VIA12_LV 2x1 R}, and {VIA12_LV 2x1 NR}.

The syntax for specifying nondefault vias using the `-via_cuts` option is

```
define_routing_rule rule_name
    -via_cuts { {via_type1 cut_number1 orientation1}
                 {via_type2 cut_number2 orientation2}
                 ...
                 {via_typen cut_numbern orientationn}
             }
```

You can specify multiple via types per layer. For each via, you must explicitly specify the allowed cut numbers and orientation. To specify the orientation, use `NR` to indicate that the via is not rotated or `R` to indicate that the via is rotated. The order of via specification is not important; during routing, Zroute selects the lowest cost nondefault via.

If you do not specify a nondefault via for a layer, Zroute selects a via from the Milkyway technology file based on the design rules.

For example, to specify the vias selected by the `-cuts` option in the previous example, enter

```
icc_shell> define_routing_rule via_rule \
    -via_cuts {{VIA12_LH 1x1 R} {VIA12_LH 1x1 NR} {VIA12_LV 1x1 R}
                {VIA12_LV 1x1 NR} {VIA12_LH 1x2 R} {VIA12_LH 1x2 NR}
                {VIA12_LH 2x1 R} {VIA12_LH 2x1 NR} {VIA12_LV 1x2 R}
                {VIA12_LV 1x2 NR} {VIA12_LV 2x1 R} {VIA12_LV 2x1 NR}}
```

Defining Shielding Rules

You can use nondefault routing rules to define shielding rules. The syntax for specifying shielding rules is

```
define_routing_rule rule_name
  -shield_widths {layer1 width1 layer2 width2 ... layern widthn}
  -shield_spacings {layer1 spacing1 layer2 spacing2 ... layern spacingn}
```

You can define a single width and spacing value per layer. Zroute uses the default wire width for any layers not specified in the `-shield_widths` option and the default spacing for any layers not specified in the `-shield_spacings` option.

By default, shielding wires are not snapped to the routing tracks. To snap shielding wires to the routing tracks, use the `-snap_to_track` option when you define the nondefault routing rule.

For example, to specify a shielding rule that uses spacing of 0.1 microns and a width of 0.1 microns for metal1 through metal5 and spacing of 0.3 microns and a width of 0.3 microns for metal6, use the following command:

```
icc_shell> define_routing_rule shield_rule \
  -shield_widths {M1 0.1 M2 0.1 M3 0.1 M4 0.1 M5 0.1 M6 0.3} \
  -shield_spacings {M1 0.1 M2 0.1 M3 0.1 M4 0.1 M5 0.1 M6 0.3}
```

Applying Nondefault Routing Rules

The IC Compiler tool 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. During clock tree synthesis, the nondefault routing rules are automatically assigned to the newly created clock nets.

- `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](#).”

For example, to assign a shielding rule called shield_rule to the nets in the CLK clock tree before clock tree synthesis, use the following command:

```
icc_shell> set_clock_tree_options -clock_trees 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 “[Specifying Clock Tree Synthesis Options](#)” on page 5-45.

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).

For example, to assign a nondefault routing rule called WideMetal to the CLK net, use the following command:

```
icc_shell> set_net_routing_rule -rule WideMetal [get_nets CLK]
```

Verifying Nondefault Routing Rules

Zroute includes violations of nondefault routing rules in the DRC summary it generates after detail routing. You can also check for DRC violations, including violations of nondefault routing rules, by using the `verify_zrt_route` command. In the DRC reports generated by Zroute, nondefault wire spacing violations are labeled as “Diff net var rule spacing” violations and nondefault via spacing violations are labeled as “Var rule via-cut spacing” violations. For more information about verifying routing rules, see “[Performing Design Rule Checking Using the IC Compiler Tool](#)” on page 6-120.

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.

Removing Nondefault Routing Rules

To remove the nondefault routing rules for specific nets, use the `set_net_routing_rule` command to reset the routing rule for the nets to the default routing rule.

```
icc_shell> set_net_routing_rule -rule default [get_nets my_ndr_nets]
```

Setting the Preferred Routing Direction

By default, IC Compiler derives the preferred routing directions from the unit tile of the first standard cell library in the reference library. The unit tile is created during library preparation, as described in the *Library Data Preparation for IC Compiler User Guide*.

To override the default preferred routing direction for a specific layer, use the `set_preferred_routing_direction` command. The layer direction set with this command applies only to the current design.

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 the M5 and M7 layers, use the following command:

```
icc_shell> set_preferred_routing_direction -layers "M5 M7" \
  -direction vertical
```

Note:

Settings made with the `create_route_guide`
`-switch_preferred_direction_layers` command, which changes the preferred
 direction within the area that is covered by the route guide, override the settings made
 with the `set_preferred_routing_direction` command.

To report the preferred routing direction for all the routing layers, use the `report_preferred_routing_direction` command. The report lists the library and

user-defined routing directions, as well as the direction that the tool will use. [Example 6-1](#) shows a report example.

Example 6-1 Preferred Direction Report

```
*****
Report : Layers
Design : core_chip
Version: D-2010.03
Date   : Thu Jan 28 03:43:06 2010
*****
Layer Name      Library      Design      Tool understands
metal1          Horizontal   Not Set    Horizontal
metal2          Vertical     Not Set    Vertical
metal3          Horizontal   Not Set    Horizontal
metal4          Vertical     Not Set    Vertical
metal5          Horizontal   Vertical   Vertical
metal6          Vertical     Vertical   Vertical
```

To remove the user-defined directions for a specific layer from the design, use the `remove_preferred_routing_direction` command. The syntax is

```
remove_preferred_routing_direction -layers list_of_layers
```

Controlling Off-Grid Routing

You can prevent off-grid routing of wires or vias by requiring them to be aligned to the wire track or discourage off-grid routing of vias by increasing the cost associated with off-grid routing.

Preventing Off-Grid Routing

By default, wires and vias need to be aligned to the wire track grid for a metal layer only if the `onWireTrack` or `onGrid` attribute is set to 1 in its `Layer` section in the technology file.

To override the technology file settings, use the `set_route_zrt_common_options` command to set the following common route options:

- `-wire_on_grid_by_layer_name`

This option controls off-grid routing for metal layers.

- `-via_on_grid_by_layer_name`

This option controls off-grid routing for via layers.

If you use either of these options, the tool ignores all settings for the `onWireTrack` and `onGrid` attributes in the technology file and uses only the settings specified by these options. If you do not specify a layer in these options, off-grid routing is allowed on that layer, regardless of the setting in the technology file.

For example, to prevent off-grid routing for wires on the M2 and M3 metal layers and for vias on the V2 via layer, regardless of the settings in the technology file, use the following commands:

```
icc_shell> set_route_zrt_common_options \
    -wire_on_grid_by_layer_name {{M2 true} {M3 true}}
icc_shell> set_route_zrt_common_options \
    -via_on_grid_by_layer_name {{V2 true}}
```

Discouraging Off-Grid Routing for Vias

To discourage off-grid routing for vias, you can increase the cost of routing the via enclosure metal shapes off the wire track grid. To specify the extra cost multiplier for the metal layers on which the via enclosures are routed, use the `set_route_zrt_common_options` command to set the `-extra_via_off_grid_cost_multiplier_by_layer_name` common route option. The cost multiplier must be a value between 0.0 and 20.0. When you specify this option, the effective cost is the base cost times (1+multiplier).

For example, assume that the technology file defines the VIA12 layer between the M1 and M2 metal layers and the VIA23 via layer between the M2 and M3 metal layers. To set the extra cost multiplier for the via enclosures on the M2 metal layer (and therefore the vias on the VIA12 and VIA23 via layers) to 0.5 (for an effective via cost of 1.5 times the base cost), use the following command:

```
icc_shell> set_route_zrt_common_options \
    -extra_via_off_grid_cost_multiplier_by_layer_name {{M2 0.5}}
```

Defining Route Guides

A route guide provides routing directives for specific areas of your design.

To create a route guide, use the `create_route_guide` command or choose Floorplan > Create Route Guide in the GUI.

Note:

Route guides defined by this command are honored by Zroute; however, they are not honored by the Advanced Route tool.

When you create a route guide, you must specify its rectangular boundary, as well as information specific to the purpose of the route guide.

To specify the boundary, use the `-coordinate` option to specify the lower-left and upper-right corners of the rectangle; you can control the snapping of the coordinates by using the `set_object_snap_type` command. When you create a route guide by using the GUI, you can type the coordinates in the dialog box or draw the rectangle in the layout view; you can also specify that the route guide should snap to the minimum grid, placement site, routing track (the default), middle routing track, or user grid.

By default, the tool names each route guide RG#*n*, where *n* is a unique integer; you can override the default name by assigning a name with the `-name` option.

Note:

The blockage, pin, and via (BPV) extraction process also creates route guides. For more information about creating route guides during BPV, see the *Library Data Preparation for IC Compiler User Guide*.

The following sections describe how to create route guides for various purposes. You can create a single route guide that serves multiple purposes.

Using Route Guides to Prevent Routing

You can use a route guide to prevent signal or preroute (power and ground) routing, including via placement, within the route guide boundary on specific layers. Zroute considers this type of route guide to be a hard constraint.

To prevent signal routing or via insertion, use the `-no_signal_layers` option to specify the layers on which signal routing or via insertion cannot occur within the route guide boundary.

To prevent preroute routing, use the `-no_preroute_layers` option to specify the layers on which preroute routing cannot occur within the route guide boundary.

By default, when you use either of these options, the router must meet the minimum spacing requirements between the route guide boundary and the net shapes. To disable the minimum spacing rule between the route guide boundary and the net shapes, use the `-zero_min_spacing` option. When you use this option, the net shapes can touch, but not overlap, the route guide boundary.

Note:

When you create a route guide to prevent via insertion, you must use the `-zero_min_spacing` option; otherwise, blockage, pin, and via extraction does not use the route guide to trim the via region.

For example, to prevent signal routing on the M1 layer within the rectangle with its lower-left corner at (0, 0) and its upper-right corner at (100, 100), use the following command:

```
icc_shell> create_route_guide -coordinate {0.0 0.0 100.0 100.0} \
    -no_signal_layers M1
```

To prevent vias on the V1 layer within the rectangle with its lower-left corner at (0, 0) and its upper-right corner at (100, 100), use the following command:

```
icc_shell> create_route_guide -coordinate {0.0 0.0 100.0 100.0} \
    -no_signal_layers V1 -zero_min_spacing
```

To prevent preroute routing on the M2 layer within the rectangle with its lower-left corner at (0, 0) and its upper-right corner at (100, 100), use the following command:

```
icc_shell> create_route_guide -coordinate {0.0 0.0 100.0 100.0} \
    -no_preroute_layers M2
```

Using Route Guides to Control the Routing Direction

You can use a route guide to control the routing direction within the route guide boundary, either by specifying the layers where routes must be in the preferred direction within the route guide boundary or by switching the preferred direction for all layers within the route guide boundary.

A route guide that forces the router to route all nets in the preferred direction within the route guide boundary is called a preferred-direction-only route guide. You can use a preferred-direction-only route guide to prevent wrong-way jog wires on specific layers. To create a preferred-direction-only route guide, use the `-preferred_direction_only_layers` option. For example, to force the router to use only the preferred direction on the M4 layer within the rectangle with its lower-left corner at (0, 0) and its upper-right corner at (100, 100), use the following command:

```
icc_shell> create_route_guide -coordinate {0.0 0.0 100.0 100.0} \
    -preferred_direction_only_layers M4
```

A route guide that switches the preferred routing direction for one or more layers within the route guide boundary is called a switch-preferred-direction route guide. If your design contains much detour routing, you might be able to reduce congestion by using a switch-preferred-direction route guide to allow routing over macros. To create a switch-preferred-direction route guide, use the `-switch_preferred_direction_layers` option. For example, to switch the preferred routing direction for the M1 and M2 layers within the rectangle with its lower-left corner at (0, 0) and its upper-right corner at (100, 100), use the following command:

```
icc_shell> create_route_guide -coordinate {0.0 0.0 100.0 100.0} \
    -switch_preferred_direction_layers {M1 M2}
```

To switch the preferred routing direction for all layers within the route guide boundary, use `all_routing_layers` as the argument to the `-switch_preferred_direction_layers` option. For example, to switch the preferred routing direction for all layers within the rectangle with its lower-left corner at (0, 0) and its upper-right corner at (100, 100), use the following command:

```
icc_shell> create_route_guide -coordinate {0.0 0.0 100.0 100.0} \
    -switch_preferred_direction_layers all_routing_layers
```

Note:

If a switch-preferred-direction route guide overlaps with a preferred-direction-only route guide, the switch-preferred-direction route guide takes precedence.

Using Route Guides to Limit Edges in the Nonpreferred Direction

You can use a route guide to limit the number of occurrences of a specific routing pattern within the route guide boundary. This type of route guide is called a maximum-number-of-pattern route guide.

To create a maximum-number-of-pattern route guide to limit the number of edges in the nonpreferred direction, use the `-max_number_of_pattern` option to define the thresholds for the `NonPrefDirEdge` pattern. Specify the threshold for each layer on which you want to limit the number of edges in the nonpreferred direction. If you do not specify a threshold for a layer, that layer has no limit on the number of edges in the nonpreferred direction.

Zroute tries to route the nets within the route guide to meet this constraint; however, if the number of edges in the nonpreferred direction exceeds the specified threshold, Zroute reports a violation.

For example, to limit the number of edges in the nonpreferred direction to two on M2 and to three on M4, with no limits on other layers, use the following command:

```
icc_shell> create_route_guide -coordinate {{50 50} {200} {200}} \
    -max_number_of_pattern {M2 2 NonPrefDirEdge M4 3 NonPrefDirEdge}
```

Using Route Guides to Control the Routing Density

You can use a route guide to control the routing density within the route guide boundary. This type of route guide is called a utilization route guide.

To create a utilization route guide, use the `-horizontal_track_utilization` option to set the maximum track utilization for layers with a horizontal preferred direction and use the `-vertical_track_utilization` option to set the maximum track utilization for layers with a vertical preferred direction. By default, when you set these options, they apply to all layers within the route guide boundary. To set the routing density only for specific layers, use the `-track_utilization_layers` option to specify the affected layers.

For example, to set a maximum track utilization of 50 percent for all layers with a horizontal preferred direction and a maximum track utilization of 30 percent for all layers with a vertical preferred direction within the rectangle with its lower-left corner at (0, 0) and its upper-right corner at (100, 100), use the following command:

```
icc_shell> create_route_guide -coordinate {0.0 0.0 100.0 100.0} \
    -horizontal_track_utilization 50 -vertical_track_utilization 30
```

Using Route Guides to Prioritize Routing Regions

You can use a route guide to prioritize regions within the route guide boundary for routing. This type of route guide is called an access preference route guide.

An access preference route guide prioritizes specific areas for routing by assigning strengths to the various regions within the route guide boundary. These regions are called

access preference areas. Access preference areas with higher strengths are preferred for routing. You can define access preference areas for both wires and vias. When you define a via access-preference area, you are defining a preference area for the via surrounds on the specified metal layer for vias coming from both above and below that metal layer.

To create an access preference route guide, use the `-access_preference` option. The syntax for specifying an access preference route guide is

```
{layer [{wire_access_preference wire_rect wire_strength}]
 [{via_access_preference via_rect via_strength}] ...}
```

You can define multiple access preference areas. To define the relative preference of the access preference areas, use strength values between 0 and 1. If access preference areas overlap, the stronger access preference area takes precedence over the weaker access preference area. To require routing in a specific access preference area, use a strength value of 1. When you define access preference areas with a strength of 1, all access preference areas with a strength less than 1 are ignored and Zroute treats the access preference route guide as a hard constraint.

The following example creates an access preference route guide whose boundary is a rectangle with its lower-left corner at (0, 0) and its upper-right corner at (300, 300). It contains one wire access-preference area with coordinates of (0, 0) and (2, 1) and two via access-preference areas, one with coordinates of (0, 0) and (5, 5) and one with coordinates of (40, 40) and (45, 45). The wire access-preference area is slightly preferred over areas outside of the access preference area because it has a strength of 0.2. The via access-preference area with coordinates at (40, 40) and (45, 45) is ignored, because routing is required in the via access-preference area with coordinates at (0, 0) and (5, 5), which has a strength of 1.

```
icc_shell> create_route_guide -coordinate {{0 0} {300 300}} \
    -access_preference {M1 {wire_access_preference {{0 0} {2 1}} 0.2}
                        {via_access_preference {{0 0} {5 5}} 1.0}
                        {via_access_preference {{40 40} {45 45}} 0.5}}
```

To report information about the access preference route guides defined for your design, use the `report_access_preference_route_guide` command.

Using Route Guides to Encourage River Routing

River routing refers to the routing of many nonshorting routes in roughly the same direction when routing is already restricted to just a few layers. To encourage river routing, create a single-layer route guide by using the `-single_layer_routing` option to specify the affected layers.

Using Route Guides to Fix Violations

When there is a violation on a preroute wire or inside a large macro that is difficult to fix, you can use a route guide to force the router to repair violations on all layers within the route guide boundary as a single switch box. To create this type of route guide, use the `-repair_as_single_sbox` option.

Querying Route Guides

To find route guides, use the `get_route_guides` command. For example, to get all the route guides in your design, enter

```
icc_shell> get_route_guides *
```

You can also find route guides by using a filter expression (`-filter expression`) or by using regions that encompass (`-within`) or touch (`-touch`) the route guides.

Removing Route Guides

To remove route guides, use the `remove_route_guide` command. You can remove a collection of route guides, such as that returned by the `get_route_guides` command; all route guides (`-all` option); or a specific named route guide (`-name` option).

Defining Routing Corridors

A routing corridor restricts Zroute global routing for specific nets to the region defined by a set of connected rectangles. In addition to specifying the region in which the routing occurs, you can also specify the minimum and maximum routing layers for each of the rectangles that comprise the routing corridor.

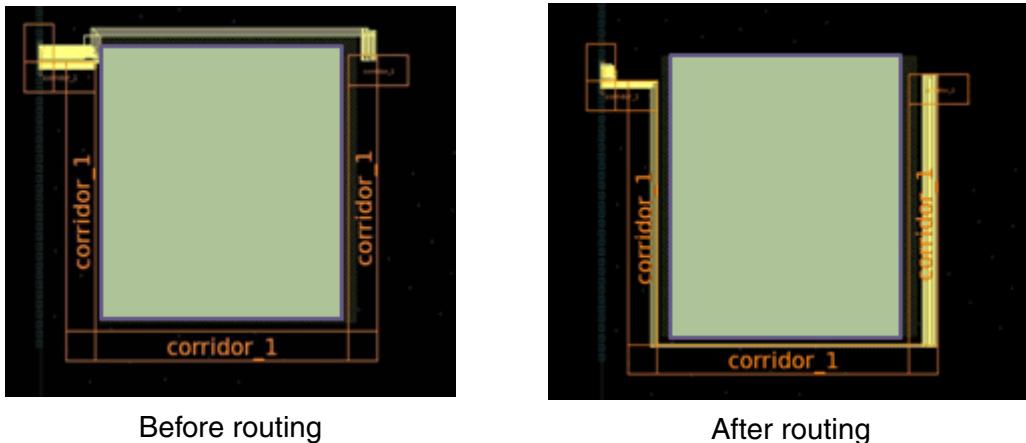
Routing corridors are intended to be used to route critical nets before signal routing. Zroute global routing considers routing corridors as a hard constraint, while track assignment and detail routing consider routing corridors as a soft constraint and might route nets slightly outside of the routing corridor to fix DRC violations.

Note:

If a route guide overlaps with a routing corridor and its attributes conflict with the routing corridor, the routing corridor takes precedence.

For example, [Figure 6-3](#) shows a routing corridor named `corridor_1`, which is made up of six rectangles. This routing corridor is associated with the nets shown in yellow. The figure on the left shows the nets before routing, while the figure on the right shows the nets routed within the routing corridor.

Figure 6-3 Using a Routing Corridor



To define a routing corridor, use the `create_routing_corridor` command (or choose Floorplan > Create Routing Corridor in the GUI). At a minimum, you must define the rectangles that comprise the routing corridor. On the command line, use the `-rectangles` option to specify the rectangles. For each rectangle, you must define the minimum routing layer, the maximum layer, and the bounding box using the following format:

```
{min_layer_name max_layer_name {bounding_box}}
```

Specify the routing layers by using the layer names from the technology file. Specify the bounding box by using the format `{lx ly urx ury}`, which defines the lower-left and upper-right corners of the rectangle in the units specified in the technology file. The tool assigns each rectangle a name in the format `CORRIDOR_SHAPE_objId`.

By default, the tool assigns each routing corridor a name in the format `CORRIDOR_objId`. You can specify a name for the routing corridor by using the `-name` option.

To assign nets to a routing corridor, use the `set_net_routing_corridor` command. Use the `-corridor` option to specify the routing corridor and the `-nets` option to specify the associated nets. You can assign a net to only one routing corridor and that routing corridor must cover all pins connected to the associated nets.

For example, to define a routing corridor named `corridor_a` and assign the nets named `n1` and `n2` to this routing corridor, use the following commands:

```
icc_shell> create_routing_corridor -name corridor_a \
    -rectangles { {M2 M4 {10 10 20 35}} \
        {M2 M4 {20 25 40 35}} \
        {M2 M5 {40 10 50 35}} }
icc_shell> set_net_routing_corridor -corridor corridor_a \
    -nets [get_nets {n1 n2}]
```

Note:

You can also assign nets to the routing corridor by using the `-nets` option when you use the `create_routing_corridor` command to create the routing corridor.

To route the nets in a routing corridor, use the `route_zrt_group` command. For more information about the `route_zrt_group` command, see “[Routing Critical Nets](#)” on [page 6-74](#). After signal routing is complete, remove the routing corridors before performing ECO routing. For information about removing routing corridors, see “[Removing Routing Corridors](#)” on [page 6-36](#).

Reporting Routing Corridors

To report the routing corridors in your design, use the `report_routing_corridors` command. You can either specify the routing corridors to report by using the `-corridors` option or you can report all routing corridors by using the `-all` option. By default, the command reports the following information for each routing corridor: its name; the rectangles associated with the routing corridor, including their names, bounding boxes, minimum routing layer, and maximum routing layer; and the nets associated with the routing corridor. To check the connectivity of the routing corridors, use the `-check_connectivity` option. To output a Tcl script that re-creates the routing corridors, use the `-output` option.

To report the routing corridor for a specific net, use the `report_net_routing_corridor` command. When you run this command, you must use the `-nets` option to specify the nets of interest.

Modifying Routing Corridors

You can make the following modifications to an existing routing corridor:

- Modify existing rectangles.

To modify existing rectangles, use the `update_routing_corridor -mode update` command. Use the `-corridor` option to specify the routing corridor that you want to update and the `-rectangles` option to specify the rectangles that you want to update. You can change the following attributes of the specified rectangles:

- The bounding box (`-bbox` option)
- The minimum routing layer (`-min_layer_name` option)
- The maximum routing layer (`-max_layer_name` option)

Each of these options takes a list of values, one value per rectangle specified in the `-rectangles` option. The format for each value is the same as when you specify that attribute when creating a routing corridor.

- Remove rectangles from the routing corridor.

To remove rectangles, use the `update_routing_corridor -mode remove` command. Use the `-corridor` option to specify the routing corridor that you want to update and the `-rectangles` option to specify the rectangles that you want to remove.

- Add new rectangles to the routing corridor.

To add new rectangles, use the `update_routing_corridor -mode add` command. Use the `-corridor` option to specify the routing corridor that you want to update. For each new rectangle, you must define the minimum routing layer (`-min_layer_name` option), the maximum routing layer (`-max_layer_name` option), and the bounding box (`-bbox` option). Each of these options takes a list of values, one value for rectangle you want to add. The format for each value is the same as when you specify that attribute when creating a routing corridor.

- Change the nets associated with the routing corridor.

To associate nets with a routing corridor, use the `set_net_routing_corridor` command. To dissociate nets with a routing corridor, use the `remove_net_routing_corridor` command.

You can also modify routing corridors in the GUI by using the Create Route Corridor tool, the Move/Resize tool, or the Delete tool, or by editing the attributes in the Properties dialog box.

Removing Routing Corridors

To remove routing corridors, use the `remove_routing_corridor` command. You can either specify the routing corridors to remove by using the `-corridors` option or you can remove all routing corridors by using the `-all` option. You can also remove routing corridors in the GUI by using the Delete tool.

Defining Routing Blockages

A routing blockage defines a rectangular or rectilinear region where no routing is allowed on a specific blockage layer. Blockage layers are system layers, not routing layers; therefore, by default, the Advanced Route tool and Zroute ignore routing blockages. The preferred method for preventing routing in a region is to use a route guide, as described in “[Using Route Guides to Prevent Routing](#)” on page 6-29.

To define a routing blockage, use the `create_routing_blockage` command (or choose Floorplan > Create Routing Blockage in the GUI).

Note:

By default, routing blockages defined by this command are ignored by both the Advanced Route tool and Zroute. To have the Advanced Route tool honor these blockages, change the setting in the Mouse Tool Options panel or the Advanced Route tool dialog box. To

have Zroute honor these routing blockages, use the `set_route_zrt_common_options` command to set the `-read_user_metal_blockage_layer` common route option to true.

To define a routing blockage, you must specify

- 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.

- 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.

For example, to create rectangular routing blockages on the `metal1` and `via1` blockage layers that are honored by Zroute, use the following commands:

```
icc_shell> create_routing_blockage \
    -layers {metal1Blockage via1Blockage} -bbox {x1 y1 x2 y2}
icc_shell> set_route_zrt_common_options \
    -read_user_metal_blockage_layer true
```

When the tool 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 *
```

You can also find routing blockages by using a filter expression (`-filter expression`), by using regions that encompass (`-within`), touch (`-touch`), or intersect (`-intersect`) the routing blockages, or by using a point overlapped by the routing blockage (`-at`).

To remove routing blockages, use the `remove_routing_blockage` command.

Controlling Pin Connections

By default, Zroute connects a signal route to a pin by using wires or vias anywhere on the pin. You can restrict the allowed types of pin connections on a per-layer basis by using the `set_route_zrt_common_options` command to set the `-connect_within_pins_by_layer_name` common route option.

Valid values for this option are

- `off` (the default)

There are no restrictions on pin connections.

- `via_standard_cell_pins`

Only the connections to standard cell pins by using a via are restricted. When using a via connection, the via's metal enclosure must be contained within the pin shape.

There are no restrictions on signal routes connected to macro cell and pad cell pins by using a via or to any pins by using wires.

- `via_wire_standard_cell_pins`

The connections to standard cell pins by using a via or a wire are restricted. When using a via connection, the via's metal enclosure must be contained within the pin shape. When using a wire, the wire must be contained within the pin shape.

- `via_all_pins`

The connections to any pins (standard cell, macro cell, or pad cell) by using a via are restricted. When using a via connection, the via's metal enclosure must be contained within the pin shape.

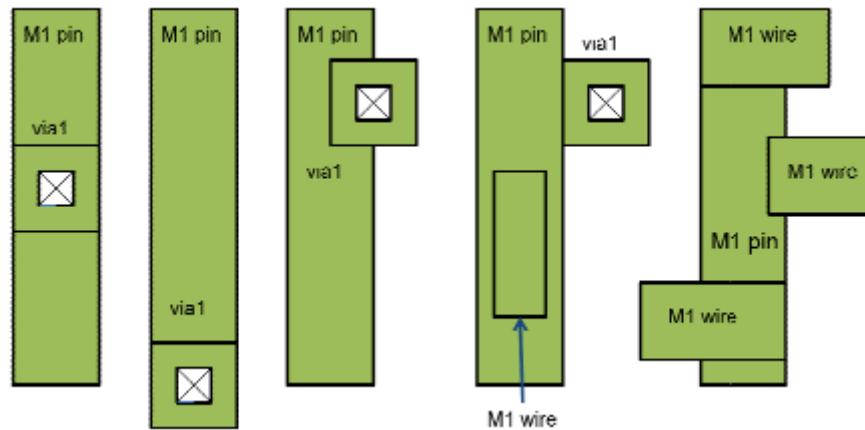
There are no restrictions on signal routes connected to any pins by using wires.

- `via_wire_all_pins`

The connections to any pins by using a via or a wire are restricted. When using a via connection, the via's metal enclosure must be contained within the pin shape. When using a wire, the wire must be contained within the pin shape.

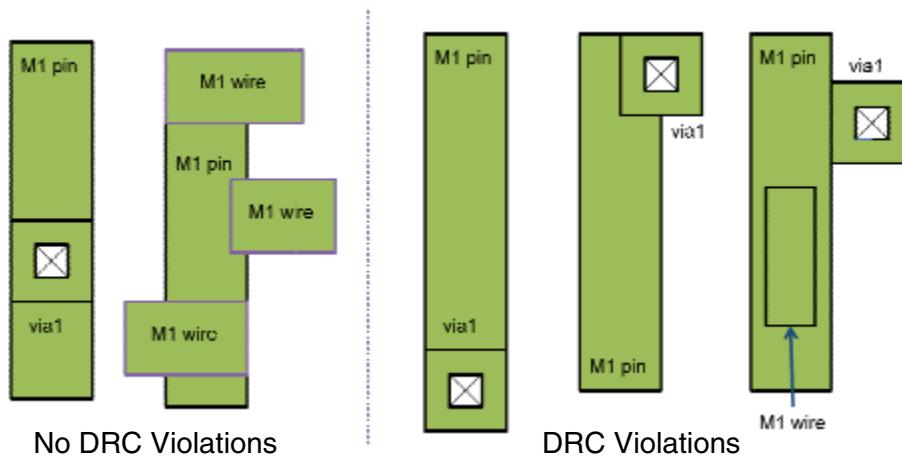
For example, if you use the following command (or use the default settings), all of the connections shown in [Figure 6-4](#) are valid and no DRC violations are reported:

```
icc_shell> set_route_zrt_common_options \
-connect_within_pins_by_layer_name {{m1 off}}
```

Figure 6-4 Unrestricted Pin Connections

If you set the mode for metal1 to `via_all_pins`, as shown in the following example, the via enclosures must be inside the pin shape. The connections shown on the left side of [Figure 6-5](#) are valid; however, the connections on the right side of the figure cause DRC violations.

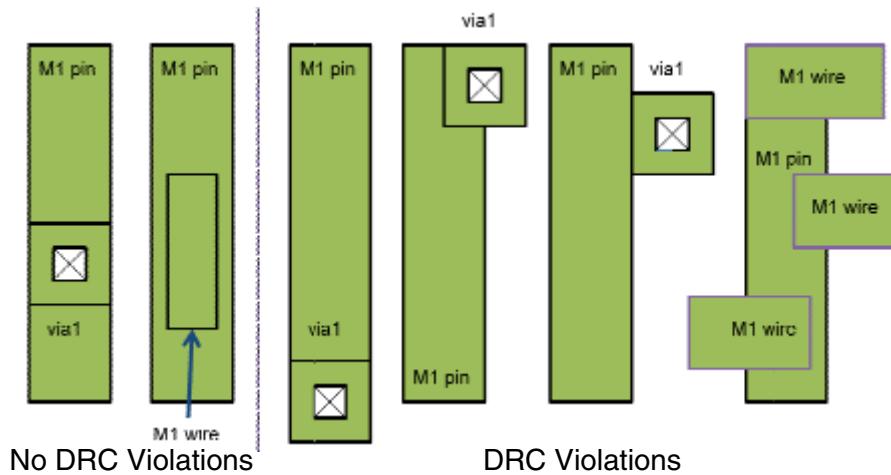
```
icc_shell> set_route_zrt_common_options \
    -connect_within_pins_by_layer_name {{m1 via_all_pins}}
```

Figure 6-5 Restricted Via-to-Pin Connections

If you set the mode for metal1 to `via_wire_standard_cell_pins`, as shown in the following example, both the via enclosures and wires must be inside the pin shape. The connections shown on the left side of [Figure 6-6](#) are valid; however the connections on the right side of the figure cause DRC violations.

```
icc_shell> set_route_zrt_common_options \
    -connect_within_pins_by_layer_name {{m1 via_wire_standard_cell_pins}}
```

Figure 6-6 Restricted Via-to-Pin and Wire-to-Pin Connections



Controlling Pin Tapering

Pin tapering is the method used to connect wires with nondefault routing rules to pins. Zroute supports pin tapering for both hard and soft nondefault routing rules; the tapering implementation is the same for both types of nondefault routing rules.

You can specify the method used for pin tapering and control the tapering width, as described in the following sections:

- [Specifying the Tapering Method](#)
- [Controlling the Tapering Width](#)

Specifying the Tapering Method

You specify the tapering method for a nondefault routing rule when you define the rule with the `define_routing_rule` command. By default, Zroute uses distance-based pin tapering; it uses the default routing rule within the tapering distance from the pin and uses the nondefault routing rule beyond the tapering distance. However, advanced process nodes are very sensitive to jogs and fat metal, and sometimes the tapering distance is not sufficient to fix the routing DRC violations on nets with nondefault routing rules. In these cases, you can use layer-based tapering, which targets DRC violations on nets with nondefault routing rules.

Note:

Distance-based tapering and layer-based tapering are mutually exclusive. If you define a routing rule that uses both a distance-based tapering option and a layer-based tapering option, the tool uses the layer-based tapering settings and ignores the distance-based tapering settings.

When Zroute performs distance-based pin tapering, it

- Determines the tapering distance, which is about 10 times the mean number of tracks for all routing layers.

To explicitly specify the tapering distance, use the `-taper_distance` option when you create a nondefault routing rule with the `define_routing_rule` command.

- Uses the same tapering distance for all pins.

To specify a different tapering distance for driver pins, use the `-driver_taper_distance` option when you create a nondefault routing rule with the `define_routing_rule` command.

To perform layer-based pin tapering, use the `-taper_over_pin_layers` or `-taper_under_pin_layers` option when you create a nondefault routing rule with the `define_routing_rule` command.

- For pins on the metal1 or metal2 layers, use the `-taper_over_pin_layers` option to specify the number of layers on or above the pin layer available for tapering. A value of 1 enables pin tapering only on the pin layer; a larger value enables pin tapering on additional layers above the pin layer.
- For pins on upper layers, use the `-taper_under_pin_layers` option to specify the number of layers on or below the pin layer available for tapering. A value of 1 enables pin tapering only on the pin layer; a larger value enables pin tapering on additional layers below the pin layer.

Controlling the Tapering Width

By default, the wire is tapered to the default routing width, which is the routing width that is defined for the metal layer in the technology file.

- To taper the wire to the pin width rather than the default routing width, use the `set_route_zrt_detail_options` command to change the `-pin_taper_mode` detail route option to `pin_width` from its default of `default_width` before you perform detail routing.
- To disable tapering for certain types of pins, use the `set_route_zrt_detail_options` command to set one or more of the following detail route options to `true`:
`-use_wide_wire_to_input_pin`, `-use_wide_wire_to_output_pin`,
`-use_wide_wire_to_macro_pin`, `-use_wide_wire_to_pad_pin`, and
`-use_wide_wire_to_port`.
- To disable tapering for all pins, set the `-taper_distance` option to 0 when you create the nondefault routing rule with the `define_routing_rule` command.

Setting the Rerouting Mode

By default, Zroute can reroute nets as needed. You can prevent rerouting or limit rerouting to minor changes by using the `-reroute` option with the `set_net_routing_rule` command.

- To prevent rerouting, use the `-reroute freeze` option.
- To limit rerouting to minor changes, use the `-reroute minorchange` option.

For example, to prevent rerouting of the `net1` net, which uses the default routing rule, use the following command:

```
icc_shell> set_net_routing_rule -rule default -reroute freeze \
[get_nets net1]
```

Setting Routing Options

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](#)
- [Setting Zroute Options](#)
- [Setting Crosstalk Prevention Options](#)
- [Setting Crosstalk Reduction 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 always get the same result. However, if you use multiple threads, the routing results are not deterministic; the final routing is 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 the IC Validator tool 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 the IC Validator tool 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 2-12.

Note:

When you enable multicore processing by using the `set_host_options` command, it enables multithreading for all IC Compiler commands that support multithreading, not just the routing commands. When multicore processing is enabled, you can limit the number of cores used by Zroute by setting the `zrt_max_parallel_computations` variable.

Setting Zroute Options

You can specify global route options, track assignment options, detail route options, and common route 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, use the `set_route_zrt_common_options` command to set the `-verbose_level` common route 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 defaults, use the `set_route_zrt_common_options -default true` command (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.

Table 6-2 lists the Zroute common route options.

Table 6-2 Common Route Options

Option	Valid values	Description
Run control options		
<code>-reroute_clock_shapes</code>	<code>true</code> <code>false</code>	Specifies whether the router can reroute fixed clock wires and vias. The default is <code>false</code> .
<code>-reroute_user_shapes</code>	<code>true</code> <code>false</code>	Specifies whether the router can reroute user-created wires and vias. The default is <code>false</code> .
<code>-plan_group_aware</code>	<code>off</code> <code>all_routing</code> <code>top_level_routing_only</code>	Specifies whether the plan group constraints are obeyed. The default is <code>off</code> .
<code>-reshield_modified_nets</code>	<code>off</code> <code>unshield</code> <code>reshield</code>	Specifies whether to unshield or reshift modified shielded nets automatically during routing. The default is <code>off</code> .

Table 6-2 Common Route Options (Continued)

Option	Valid values	Description
<code>-child_process_net_threshold</code>	<i>int</i> (must be between -1 and 2147483647)	Specifies the threshold number of nets to trigger a separate router process. The default is -1, which means that the tool decides when to trigger a separate router process based on the number of nets in the design and the peak memory of the parent process when the routing command is run.
<code>-verbose_level</code>	0 1	Sets the verbosity level for the routing log file. The default is 0.
<code>-clock_topology</code>	comb normal	Specifies the clock routing topology. The default is <code>normal</code> .
<code>-comb_distance</code>	<i>int</i> (must be between 0 and 50)	Specifies the number of global routing cells within which to connect clock pins to the clock mesh. The default is 2.
<code>-pg_shield_distance_threshold</code>	<i>float</i>	Specifies the distance threshold in microns for the router to consider the existing power and ground structure as shielding when calculating the shielding ratio. The default is 0.
<code>-number_of_secondary_pg_pin_connections</code>	<i>int</i>	Specifies the maximum number of connected secondary power or ground (PG) pins that have a single connection to a PG strap or ring.

Table 6-2 Common Route Options (Continued)

Option	Valid values	Description
<code>-separate_tie_off_from_secondary_pg</code>	<code>true false</code>	Specifies whether to separate secondary power and ground connections from tie-off connections. The default is <code>false</code> .
<code>-connect_floating_shapes</code>	<code>true false</code>	Specifies whether to connect floating shapes in addition to the pins. The default is <code>false</code> .
<code>-allow_pg_as_shield</code>	<code>true false</code>	Specifies whether the router should treat power and ground nets as shields. The default is <code>true</code> .
<code>-shielding_nets</code>	<code>pg_nets</code>	Specifies the power and ground nets to be used for shielding by the <code>create_zrt_shield</code> command.
<code>-rc_driven_setup_effort_level</code>	<code>off low medium high</code>	Controls the extent to which timing-driven routing prefers lower-resistance metal layers for routing timing-critical nets. The default is <code>medium</code> .
<code>-ignore_var_spacing_to_blockage</code>	<code>true false</code>	Controls whether nondefault routing rules are ignored against blockages. The default is <code>false</code> .
<code>-ignore_var_spacing_to_pg</code>	<code>true false</code>	Controls whether nondefault routing rules are ignored against all wires on PG nets. The default is <code>false</code> .
<code>-ignore_var_spacing_to_shield</code>	<code>true false</code>	Controls whether nondefault routing rules are ignored against shield wires on PG nets. The default is <code>true</code> .

Table 6-2 Common Route Options (Continued)

Option	Valid values	Description
Boundary and blockage options		
-read_user_metal_blockage_layer	true false	Specifies whether shapes on metal blockage layers should be honored (true) or ignored (false). The default is false.
-wide_macro_pin_as_fat_wire	true false	Specifies whether or not the wide macro or pad pin should have an implicit depth same as its width. The default is false.
-route_top_boundary_mode	stay_half_min_space_inside stay_inside ignore	Controls the routing behavior near the top boundary. The default is stay_inside.
-standard_cell_blockage_as_thin	true false	Specifies whether or not to treat wide blockages in standard cells as thin wires. The default is false.
Layer options		
-freeze_layer_by_layer_name	{{layer true false}...}	Controls whether routing layers are frozen to prevent them from changing during routing. By default (false), routing layers are not frozen.
-freeze_via_to_frozen_layer_by_layer_name	{{layer true false}...}	Controls the treatment of vias that touch a frozen layer on one side. The default is false for all layers.

Table 6-2 Common Route Options (Continued)

Option	Valid values	Description
<code>-forbid_new_metal_by_layer_name</code>	<code>{{layer true false}...}</code>	Controls whether ECO routing can remove, but not add, metal on the specified layers.
		For all other routing commands, this option acts the same as the <code>freeze_layer_by_layer_name</code> option and controls whether routing cannot add or remove metal on the specified layers.
		The default is <code>false</code> for all layers.
<code>-net_max_layer_mode</code>	<code>soft allow_pin_ connection hard</code>	Specifies the strength of the net-specific maximum layer constraint.
		The default is <code>hard</code> .
<code>-net_max_layer_mode_soft_cost</code>	<code>low medium high</code>	Specifies the cost of violating soft net-specific maximum layer constraints.
		The default is <code>medium</code> .
<code>-net_min_layer_mode</code>	<code>soft allow_pin_ connection hard</code>	Specifies the strength of the net-specific minimum layer constraint.
		The default is <code>soft</code> .
<code>-net_min_layer_mode_soft_cost</code>	<code>low medium high</code>	Specifies the cost of violating soft net-specific minimum layer constraints.
		The default is <code>medium</code> .
<code>-global_max_layer_mode</code>	<code>soft allow_pin_ connection hard</code>	Specifies the strength of the global maximum layer constraint.
		The default is <code>soft</code> .
<code>-global_min_layer_mode</code>	<code>soft allow_pin_ connection hard</code>	Specifies the strength of the global minimum layer constraint.
		The default is <code>soft</code> .

Table 6-2 Common Route Options (Continued)

Option	Valid values	Description
<code>-extra_preferred_direction_wire_cost_multiplier_by_layer_name</code>	<code>\{{layer float}\}...</code>	Specifies the cost multiplier for routing in the preferred direction.
<code>-extra_nonpreferred_direction_wire_cost_multiplier_by_layer_name</code>	<code>\{{layer float}\}...</code>	Specifies the cost multiplier for routing in the nonpreferred direction.
<code>-extra_via_off_grid_cost_multiplier_by_layer_name</code>	<code>\{{layer float}\}...</code>	Specifies the extra cost multiplier for routing the metal surround of a via off the wire track. The multiplier is specified for a metal layer and affects the adjacent via layers.

Via options

<code>-number_of_vias_over_net_max_layer</code>	<i>int</i> (must be between 0 and 15)	Specifies the maximum level of stacked vias that can be used to access pins or fixed routes above the net-specific maximum routing layer. The default is 1.
<code>-number_of_vias_under_net_min_layer</code>	<i>int</i> (must be between 0 and 15)	Specifies the maximum level of stacked vias that can be used to access pins or fixed routes below the net-specific minimum routing layer. The default is 1.

Table 6-2 Common Route Options (Continued)

Option	Valid values	Description
<code>-number_of_vias_over_global_max_layer</code>	<i>int</i> (must be between 0 and 15)	Specifies the maximum level of stacked vias that can be used to access pins or fixed routes above the global maximum routing layer. The default is 1.
<code>-number_of_vias_under_global_min_layer</code>	<i>int</i> (must be between 0 and 15)	Specifies the maximum level of stacked vias that can be used to access pins or fixed routes below the global minimum routing layer. The default is 1.
<code>-via_array_mode</code>	<code>off swap rotate all</code>	Specifies the types of rotated via arrays that can be used during signal routing and redundant via insertion. The default is <code>all</code> . Zroute uses this option only when routing with default routing rules. For routing with nondefault routing rules, you must explicitly specify all supported via configurations by using the <code>define_routing_rule</code> command.
<code>-post_detail_route_redundant_via_insertion</code>	<code>off low medium high</code>	Enables automatic redundant via insertion after detail routing and ECO routing. The default is <code>off</code> .
<code>-concurrent_redundant_via_mode</code>	<code>off reserve_space insert_at_high_cost</code>	Enables concurrent redundant via insertion during initial routing. The default is <code>off</code> .
<code>-concurrent_redundant_via_effort_level</code>	<code>low medium high</code>	Specifies the effort level for concurrent redundant via insertion during initial routing. The default is <code>low</code> .

Table 6-2 Common Route Options (Continued)

Option	Valid values	Description
-eco_route_concurrent_redundant_via_mode	off reserve_space	Enables concurrent redundant via insertion during ECO routing. The default is off.
-eco_route_concurrent_redundant_via_effort_level	low medium high	Specifies the effort level for concurrent redundant via insertion during ECO routing. The default is low.
-rotate_default_vias	true false	Specifies whether the router can rotate default vias. The default is true.
-extra_via_cost_multiplier_by_layer_name	{layer float}...	Specifies the cost multiplier for vias.
-via_on_grid_by_layer_name	{layer true false}...	Controls whether off-grid routing is allowed on the specified via layers.

Soft rule options

-route_soft_rule_effort_level	off min low medium high	Specifies the default effort level that is used to rip up and reroute DRC violations for soft spacing rules. The default is medium.
-post_detail_route_fix_soft_violations	true false	Enables the fixing of soft rule violations, such as bridge rules, after detail routing. This option does not affect the fixing of soft spacing rules. The default is false.
-post_group_route_fix_soft_violations	true false	Enables the fixing of soft rule violations, such as bridge rules, after group routing. This option does not affect the fixing of soft spacing rules. The default is false.

Table 6-2 Common Route Options (Continued)

Option	Valid values	Description
<code>-post_eco_route_fix_soft_violations</code>	<code>true false</code>	Enables the fixing of soft rule violations, such as bridge rules, after ECO routing. This option does not affect the fixing of soft spacing rules. The default is <code>false</code> .
<code>-post_incremental_detail_route_fix_soft_violations</code>	<code>true false</code>	Enables the fixing of soft rule violations, such as bridge rules, after incremental detail routing. This option does not affect the fixing of soft spacing rules. The default is <code>false</code> .
<code>-routing_rule_effort_level</code>	<code>rule_effort_list</code>	Specifies the effort level that is used to rip up and reroute DRC violations for the specified soft spacing routing rules.
<code>-soft_rule_weight_to_effort_level_map</code>	<code>\{{weight_effort}\}...</code>	Specifies a global mapping from soft weight level to soft effort level.

Voltage area options

<code>-enforce_voltage_areas</code>	<code>off strict relaxed</code>	Specifies the level of enforcement for voltage area constraints. The default is <code>relaxed</code> .
<code>-voltage_area_weight</code>	<code>\{{voltage_area_weight}\}...</code>	Specifies the weighting for each voltage area. By default, all voltage areas have a weight of medium.

Table 6-2 Common Route Options (Continued)

Option	Valid values	Description
Miscellaneous options		
-single_connection_to_pins	off standard_cell_pins all_pins	Specifies whether the router can connect to a pin only one time. The default is off, except for nets that use a nondefault width routing rule. For those nets, the value is always all_pins.
-mark_clock_nets_minor_change	true false	Specifies whether only minor changes can be made to clock nets. The default is true.
-track_auto_fill	true false	Specifies whether to fill empty space with routing tracks. The default is true.
-report_local_double_pattern_odd_cycles	true false	Specifies whether to report local odd-cycle violations. For information about this option, see the <i>IC Compiler Advanced Geometries User Guide</i> . The default is false.
-tie_off_mode	all rail_only	Controls whether the router can connect tie-off nets to any power or ground (PG) structure, such as straps, rails, or pins or only to a PG rail. The default is all.
-connect_within_pins_by_layer_name	{ {layer off via_standard_cell_pins via_wire_standard_cell_pins via_all_pins via_wire_all_pins} ... }	Specifies which pin connections must be made within the pins for the specified layer. By default, this option is off for all layers.

Table 6-2 Common Route Options (Continued)

Option	Valid values	Description
<code>-min_edge_offset_for_macro_pin_connection_by_layer_name</code>	<code> {{layer offset}...}</code>	Overrides the minimum length specified by the <code>minEdgeOffsetForMacroPinConnection</code> attribute in the technology file for the specified layers.
<code>-wire_on_grid_by_layer_name</code>	<code> {{layer true false}...}</code>	Controls whether off-grid routing is allowed on the specified layers.

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 defaults, use the `set_route_zrt_global_options -default true` command (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.

[Table 6-3](#) lists the Zroute global route options.

Table 6-3 Global Route Options

Option	Valid values	Description
Run control options		
<code>-timing_driven</code>	<code>true false</code>	Enables (<code>true</code>) or disables (<code>false</code>) timing-driven global routing. The default is <code>false</code> .
<code>-timing_driven_effort_level</code>	<code>low medium high</code>	Specifies the effort level for the timing-driven flow. The default is <code>high</code> .

Table 6-3 Global Route Options (Continued)

Option	Valid values	Description
<code>-crosstalk_driven</code>	<code>true</code> <code>false</code>	Enables (<code>true</code>) or disables (<code>false</code>) crosstalk-driven global routing. The default is <code>false</code> .
<code>-layer_based_congestion_map</code>	<code>true</code> <code>false</code>	If <code>true</code> (the default), create a congestion map for each layer in addition to the aggregate congestion map. If <code>false</code> , create only the aggregate congestion map for the design. The default is <code>true</code> .
<code>-exclude_blocked_gcells_from_congestion_report</code>	<code>true</code> <code>false</code>	Controls whether blocked global routing cells (gcells) are included in the overflow percentage calculation. The default is <code>false</code> .
<code>-effort</code>	<code>minimum</code> <code>low</code> <code>medium</code> <code>high</code> <code>ultra</code>	Specifies the global route effort. The default is <code>medium</code> .
<code>-force_full_effort</code>	<code>true</code> <code>false</code>	If <code>false</code> (the default), the global router might stop before the maximum number of iterations, based on the incremental congestion reduction. If <code>true</code> , the global router performs the maximum number of iterations based on the specified effort level.
<code>-double_pattern_utilization_by_layer_name</code>	{ <i>layer utilization</i> }	Specifies the double-patterning utilization for per layer. For information about this option, see the <i>IC Compiler Advanced Geometries User Guide</i> . The default is 100.0.

Table 6-3 Global Route Options (Continued)

Option	Valid values	Description
Macro options		
-macro_boundary_track_utilization	<i>int</i> (must be between 0 and 100)	Specifies the percentage of tracks to be used along macro boundaries. The default is 100.
-macro_boundary_width	<i>int</i> (must be between 0 and 10)	Specifies the width of the macro boundary in terms of global routing cells. The default is 5.
-macro_corner_track_utilization	<i>int</i> (must be between 0 and 100)	Specifies the percentage of tracks to be used along macro corners. The default is 100.
Miscellaneous options		
-extra_blocked_layer_utilization_reduction	<i>int</i> (must be between 0 and 100)	Specifies an additional utilization reduction to account for the additional routing resources that might be required over potentially difficult-to-route areas during track assignment and detail routing. The default is 0.
-voltage_area_corner_track_utilization	<i>int</i> (must be between 0 and 100)	Specifies the percentage of tracks to be used along voltage area boundaries. The default is 100. Note that you must set this option from the command line; there is no GUI support for this option.

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 defaults, use the `set_route_zrt_track_options -default true` command (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.

[Table 6-4](#) lists the Zroute track assignment options.

Table 6-4 Track Assignment Options

Option	Valid values	Description
<code>-timing_driven</code>	<code>true</code> <code>false</code>	Enables (<code>true</code>) or disables (<code>false</code>) timing-driven track assignment. The default is <code>false</code> .
<code>-crosstalk_driven</code>	<code>true</code> <code>false</code>	Enables (<code>true</code>) or disables (<code>false</code>) crosstalk-driven track assignment. The default is <code>false</code> .

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 defaults, use the `set_route_zrt_detail_options -default true` command (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_options` command.

Table 6-5 lists the Zroute detail route options.

Table 6-5 Detail Route Options

Option	Valid values	Description
Run control options		
-timing_driven	true false	Enables (true) or disables (false) timing-driven routing. The default is false.
-eco_route_use_soft_spacing_for_timing_optimization	true false	Enables (true) or disables (false) soft-rule-based timing optimization during ECO routing. The default is true.
-continue_after_large_design_rule_value_error	true false	Specifies whether detail routing fails if there are large design rule values. The default is false.
-repair_shorts_over_macros_effort_level	off low medium high	Specifies the effort level used for repairing shorts over macros. The default is off.
-optimize_wire_via_effort_level	off low medium high	Specifies the effort level used to optimize wire length and via counts. The default is low.
-optimize_tie_off_effort_level	off low high	Specifies the effort level used to optimize wire length and via counts for tie-off nets. The default is low.
-generate_extra_off_grid_pin_tracks	true false	Specifies whether to generate extra off-grid routing tracks for pin connections. The default is false.
-generate_off_grid_feed_through_tracks	off low medium high	Specifies the effort level used to generate extra off-grid routing tracks for feedthrough nets between blockages. The default is low.

Table 6-5 Detail Route Options (Continued)

Option	Valid values	Description
<code>-save_after_iterations</code>	<code>list_of_ints</code>	Specifies the iterations after which to save the design. The default is {}.
<code>-save_cell_prefix</code>	<code>string</code>	Specifies the prefix to use for intermediate saved designs. The default is <i>DR</i> .
<code>-force_max_number_iterations</code>	<code>true false</code>	Controls whether the maximum number of iterations must be run when design rule checking does not converge. The default is <code>false</code> .
<code>-drc_convergence_effort_level</code>	<code>low medium high</code>	Controls the effort level for performing additional detail routing iterations before early termination when design rule checking does not converge. The default is <code>medium</code> .
<code>-user_defined_partition</code>	<code>{llx lly urx ury}</code>	Specifies the coordinates, in database units, of a rectangular partition to be added for routing.
<code>-cpu_limit</code>	<code>int (must be between -1 and 2147483647)</code>	Specifies the limit for CPU time in minutes. The default is -1, which means there is no limit.
<code>-check_patchable_drc_from_fixed_shapes</code>	<code>true false</code>	Specifies whether to check for and fix end-of-line spacing, minimum area, and minimum length violations for fixed routes. The default is <code>false</code> .

Table 6-5 Detail Route Options (Continued)

Option	Valid values	Description
<code>-report_ignore_drc</code>	<i>list_of_drc_types</i>	<p>Enables an additional DRC report and specifies the DRC types to be suppressed from this report. The additional report is printed after the DRC summary and includes all DRC violations, except internal-only types and those specified in the list.</p> <p>By default, the additional DRC report is not generated.</p>

Antenna options

<code>-antenna</code>	<code>true false</code>	Enables (<code>true</code>) or disables (<code>false</code>) antenna analysis. The default is <code>true</code> .
<code>-antenna_on_iteration</code>	<code>int</code> (must be between 1 and 19)	Specifies the routing iteration at which antenna analysis and optimization is turned on. The default is 1.
<code>-check_antenna_on_pg</code>	<code>true false</code>	Enables (<code>true</code>) or disables (<code>false</code>) analysis and optimization of antennas on power and ground nets. The default is <code>false</code> .
<code>-default_diode_protection</code>	<code>float</code> (must be between 0.00 and 1e+06)	Specifies the diode protection value used for pins during antenna analysis if the value is not specified in the cell. The default is 0.00.
<code>-default_gate_size</code>	<code>float</code> (must be between 0.00 and 1e+06)	Specifies the gate size used for pins during antenna analysis if the gate size is not specified in the cell. The default is 0.00.

Table 6-5 Detail Route Options (Continued)

Option	Valid values	Description
<code>-default_port_external_antenna_area</code>	<i>float</i> (must be between 0.00 and 1e+06)	Specifies the antenna area value used for ports (top-level pins) during antenna analysis if the antenna area is not specified in the cell. The default is 0.00.
<code>-default_port_external_gate_size</code>	<i>float</i> (must be between 0.00 and 1e+06)	Specifies the gate size used for ports (top-level pins) during antenna analysis if the gate size is not specified in the cell. The default is 0.00.
<code>-diode_libcell_names</code>	<i>list_of_libcells</i>	Specifies the diode library cells to use for antenna fixing. By default, Zroute selects the diode cells from the library.
<code>-hop_layers_to_fix_antenna</code>	<code>true false</code>	Specifies whether layers can be hopped to fix antenna violations. The default is <code>true</code> .
<code>-insert_diodes_during_routing</code>	<code>true false</code>	Specifies whether the router can use diode insertion to fix antenna violations. The default is <code>false</code> .
<code>-antenna_fixing_preference</code>	<code>hop_layers use_diodes</code>	Specifies the preferred method for fixing antenna violations. The default is <code>hop_layers</code> .
<code>-diode_insertion_mode</code>	<code>new_and_spare new spare</code>	Specifies whether the router should insert new diodes, reuse existing spare diodes, or use the closest solution. This option has an effect only when the <code>insert_diodes_during_routing</code> option is <code>true</code> . The default is to use the closest solution (<code>new_and_spare</code>).

Table 6-5 Detail Route Options (Continued)

Option	Valid values	Description
<code>-diode_preference</code>	<code>none new spare</code>	<p>Specifies whether the router should insert new diodes, reuse existing spare diodes, or use the closest solution.</p> <p>This option has an effect only when the <code>diode_insertion_mode</code> option is <code>new_and_spare</code>.</p> <p>The default is to use the closest solution (<code>none</code>).</p>
<code>-reuse_filler_locations_for_diodes</code>	<code>true false</code>	<p>Specifies whether to reuse filler cell locations for inserting diodes.</p> <p>The default is <code>true</code>.</p>
<code>-max_antenna_pin_count</code>	<code>int</code> (must be between -1 and 1000000)	<p>Specifies the maximum number of pins on a net on which antenna checking is performed.</p> <p>The default is -1, which means there is no limit.</p>
<code>-skip_antenna_fixing_for_nets</code>	<code>nets</code>	<p>Disables antenna fixing for the specified nets. Zroute performs antenna analysis on these nets and reports violations, but does not fix them.</p>
<code>-merge_gates_for_antenna</code>	<code>true false</code>	<p>Enables (<code>true</code>) or disables (<code>false</code>) merging gates for antenna analysis.</p> <p>The default is <code>true</code>.</p>
<code>-macro_pin_antenna_mode</code>	<code>normal jump top_layer</code>	<p>Specifies how the macro pins are treated for antenna consideration.</p> <p>The default is <code>normal</code>.</p>
<code>-port_antenna_mode</code>	<code>float jump top_layer</code>	<p>Specifies how the ports (top-level pins) are treated for antenna consideration.</p> <p>The default is <code>float</code>.</p>

Table 6-5 Detail Route Options (Continued)

Option	Valid values	Description
<code>-top_layer_antenna_fix_threshold</code>	<i>int</i> (must be between -1 and 10)	Specifies the threshold for fixing antenna violations on the top routing layer. The default is -1, which means there is no limit.
<code>-antenna_verbose_level</code>	0 1	Sets the antenna checking verbosity level in the routing log file. The default is 1.

Pin and port options

<code>-check_pin_min_area_min_length</code>	true false	Enables (true) or disables (false) checking for minimum area and minimum length rules on pins. The default is true.
<code>-check_port_min_area_min_length</code>	true false	Enables (true) or disables (false) checking for minimum area and minimum length rules on ports. The default is true.
<code>-pin_taper_mode</code>	default_width pin_width off	Specifies the pin tapering mode. The default is default_width.
<code>-use_wide_wire_to_input_pin</code>	true false	Specifies whether pin tapering to input pins is disallowed. The default is false.
<code>-use_wide_wire_to_macro_pin</code>	true false	Specifies whether pin tapering to macro pins is disallowed. The default is false.
<code>-use_wide_wire_to_output_pin</code>	true false	Specifies whether pin tapering to output pins is disallowed. The default is false.

Table 6-5 Detail Route Options (Continued)

Option	Valid values	Description
-use_wide_wire_to_pad_pin	same_as_macro_pin true false	Controls pin tapering to pad pins. The default is same_as_macro_pin.
-use_wide_wire_to_port	same_as_macro_pin true false	Controls pin tapering to ports. The default is same_as_macro_pin.
-use_lower_hierarchy_for_port_diodes	true false	Controls the logic hierarchy in which diodes are inserted to fix antenna violations of physical ports (terminals). By default (false), the diodes are inserted at the top level.

Width and spacing options

-diagonal_min_width	true false	Specifies whether to use diagonal or Manhattan distance for minimum width violations. By default (true), diagonal distance is used.
-ignore_drc	{ {same_net_metal_space same_net_enclosed_cut_space all_same_net_drc_for_frozen_net true false} ... }	Controls whether the router ignores specific design rule checks (DRCs). By default (false), none of the DRCs are ignored.
-use_default_width_for_min_area_min_len_stub	true false	Specifies whether to use default width stubs to fix minimum area and minimum length violations. The default is false.
-var_spacing_to_same_net	true false	Specifies if nondefault spacing is to be applied to shapes of the same net. The default is false.

Table 6-5 Detail Route Options (Continued)

Option	Valid values	Description
Design rule options		
<code>-detail_route_special_design_rule_fixing_stage</code>	<code>none early_routing late_routing</code>	Specifies the stage at which to fix special design rules during detail routing. The default is <code>late_routing</code> .
<code>-eco_route_special_design_rule_fixing_stage</code>	<code>none early_routing late_routing</code>	Specifies the stage at which to fix special design rules during ECO routing. The default is <code>late_routing</code> .
<code>-group_route_special_design_rule_fixing_stage</code>	<code>none early_routing late_routing</code>	Specifies the stage at which to fix special design rules during group routing. The default is <code>late_routing</code> .
<code>-post_process_special_design_rule_fixing_stage</code>	<code>none early_routing late_routing</code>	Specifies the stage at which to fix special design rules during post processing. The default is <code>late_routing</code> .
<code>-incremental_detail_route_special_design_rule_fixing_stage</code>	<code>none early_routing late_routing</code>	Specifies the stage at which to fix special design rules during incremental detail routing. The default is <code>late_routing</code> .

Setting Crosstalk Prevention Options

By default, Zroute does not perform crosstalk prevention. To enable crosstalk prevention during global routing and track assignment, use the following command:

```
icc_shell> set_si_options -route_xtalk_prevention true
```

When you run this command, the tool automatically sets the Zroute global route and track assignment `crosstalk_driven` options to `true`.

The default crosstalk prevention threshold is 0.35 volts, which is probably too relaxed. If your design has crosstalk violations, 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 enable crosstalk prevention, Zroute 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.

The IC Compiler tool can also perform crosstalk fixing and consider signal integrity during postroute optimization. For details, see “[Setting Crosstalk Reduction Options](#)” on page 6-66.

For more information about signal integrity, see [Chapter 8, “Signal Integrity.”](#)

Setting Crosstalk Reduction Options

To perform crosstalk reduction, use the `route_opt` core command; the atomic routing commands, such as `route_zrt_group` and `route_zrt_detail` do not perform crosstalk reduction.

By default, the `route_opt` command does not perform crosstalk reduction. To perform crosstalk reduction,

1. Enable signal integrity mode by setting the `set_si_options -delta_delay` option to `true`.

When you enable signal integrity mode, the `route_opt` command also considers signal integrity during postroute optimization.

2. (Optional) Enable static noise fixing by setting the `set_si_options -static_noise` option to `true`.

When you enable static noise fixing, the `route_opt` command also fixes static noise violations during postroute optimization.

3. Run the `route_opt` command with the `-xtalk_reduction` or `-only_xtalk_reduction` option.

To perform both crosstalk reduction and postroute optimization, use the `-xtalk_reduction` option. To perform only crosstalk reduction and not postroute optimization, use the `-only_xtalk_reduction` option.

By default, when signal integrity mode is enabled and you run the `route_opt` command with the `-xtalk_reduction` or `-only_xtalk_reduction` option, Zroute performs a single iteration of crosstalk reduction on all nets with setup violations and, if enabled, static noise violations. You can increase the number of iterations by setting the `-xtalk_reduction_loops` option with the `set_route_opt_strategy` command.

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.

The default method for reducing crosstalk is to increase the spacing between nets. To further reduce crosstalk and improve timing QoR, the tool can perform cell sizing by using footprint swapping on cells on the victim or aggressor nets in addition to increasing the spacing between nets. To enable cell sizing during crosstalk reduction, set the `routeopt_xtalk_reduction_cell_sizing` variable to `true` before running the `route_opt` command.

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.

Setup-Based Net Selection

By default, Zroute performs crosstalk reduction to fix the setup violations on all violating nets.

You can limit the number of nets on which setup fixing is performed by

- Setting a slack threshold (`-setup_slack_threshold` option)
When you use this option, Zroute performs setup fixing only on those nets with a setup slack value that is worse than the specified threshold.
- Setting an individual delta delay threshold (`-setup_one_net_delta_delay_threshold` option)
When you use this option, Zroute performs setup fixing only on those nets with setup violations and a delta delay value that exceeds the specified threshold.
- Setting a total delta delay threshold (`-setup_total_delta_delay_threshold` option)
When you use this option and the total delta delay of all nets with setup violations exceeds the specified threshold, Zroute performs setup fixing only on the nets with the top contributions to the threshold delta delay value.
- Setting a maximum net count (`-setup_max_net_count` option)
When you use this option, Zroute performs setup fixing on, at most, the specified number of nets. If the number of nets with setup violations exceeds the specified number, Zroute performs setup fixing on the nets with the worst slack violations.
- Setting a minimum net count (`-setup_min_net_count` option)
When you use this option, Zroute performs setup fixing only if the number of nets with setup violations is greater than the specified number.

You can disable setup fixing by setting the `-setup` option to `false`.

Hold-Based Net Selection

By default, Zroute does not perform crosstalk reduction to fix hold violations. To enable hold fixing, set the `-hold` option to `true`.

You can limit the number of nets on which hold fixing is performed by

- Setting a slack threshold (`-hold_slack_threshold` option)

When you use this option, Zroute performs hold fixing only on those nets with a hold slack value that is worse than the specified threshold.

- Setting a delta delay threshold (`-hold_one_net_delta_delay_threshold` option)

When you use this option, Zroute performs hold fixing only on those nets with hold violations and a delta delay value that exceeds the specified threshold.

- Setting a maximum net count (`-hold_max_net_count` option)

When you use this option, Zroute performs hold fixing on at most the specified number of nets. If the number of nets with hold violations exceeds the specified number, Zroute performs hold fixing on the nets with the worst slack violations.

- Setting a minimum net count (`-hold_min_net_count` option)

When you use this option, Zroute performs hold fixing only if the number of nets with hold violations is greater than the specified number.

Transition-Based Net Selection

By default, Zroute does not perform crosstalk reduction to fix transition time violations. To enable transition time fixing, set the `-transition` option to `true`.

You can limit the number of nets on which transition time fixing is performed by

- Setting a slack threshold (`-transition_slack_threshold` option)

When you use this option, Zroute performs transition time fixing only on those nets with a transition time slack value that is worse than the specified threshold.

- Setting a delta delay threshold (`-transition_one_net_delta_delay_threshold` option)

When you use this option, Zroute performs transition time fixing only on those nets with transition time violations and a delta delay value that exceeds the specified threshold.

- Setting a maximum net count (`-transition_max_net_count` option)

When you use this option, Zroute performs transition time fixing on at most the specified number of nets. If the number of nets with transition time violations exceeds the specified number, Zroute performs transition time fixing on the nets with the worst slack violations.

- Setting a minimum net count (`-transition_min_net_count` option)

When you use this option, Zroute performs transition time fixing only if the number of nets with transition time violations is greater than the specified number.

Static-Noise-Based Net Selection

By default, Zroute performs crosstalk reduction to fix static noise violations on all violating nets when the `set_si_options -static_noise` option is true.

Note:

When the `set_si_options -static_noise` option is false, Zroute does not perform static noise fixing, regardless of the `route_opt` crosstalk reduction settings.

You can limit the number of nets on which static noise fixing is performed by

- Setting a maximum net count (`-static_noise_max_net_count` option)

When you use this option, Zroute performs static noise fixing on at most the specified number of nets. If the number of nets with static noise violations exceeds the specified number, Zroute performs static noise fixing on the nets with the worst violations.

- Setting a minimum net count (`-static_noise_min_net_count` option)

When you use this option, Zroute performs static noise fixing only if the number of nets with static noise violations is greater than the specified number.

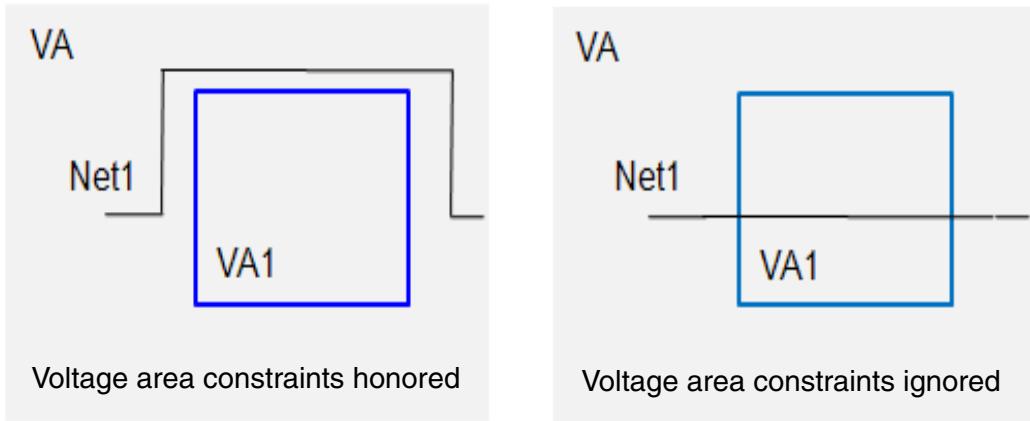
Setting Multivoltage Options

A voltage area is a placement area for one or more logic partitions that operate at the same voltage. The cells of the logic partition are associated with the partition's voltage area and are constrained to placement within that area.

By default, Zroute considers voltage area constraints during global routing, track assignment, and detail routing and tries to route wires within their native voltage area; however, it can route through non-native voltage areas to resolve congestion problems or DRC violations.

[Figure 6-7](#) shows an example of routing a net in a multivoltage design. In the figure on the left, Zroute honors the voltage area constraints and routes around the non-native voltage area. In the figure on the right, Zroute ignores the voltage area constraints and routes through the non-native voltage area.

Figure 6-7 Routing a Net in a Multivoltage Design



You can control the level of enforcement of voltage area constraints by using the `set_route_zrt_common_options` command to set the `-enforce_voltage_areas` common route option. This option supports the following values:

- `relaxed` (the default)
Global routing, track assignment, and detail routing can violate the voltage area constraints to resolve congestion or DRC issues.
- `strict`
Global routing, track assignment, and detail routing strictly follow the voltage area constraints.
- `off`
Zroute ignores the voltage area constraints.

For example, to enable strict enforcement of the voltage area constraints, use the following command:

```
icc_shell> set_route_zrt_common_options -enforce_voltage_areas strict
```

For critical nets, it might be necessary to ignore the voltage area constraints to achieve the best routing solution, even when voltage area constraints are enforced for the design. To ignore the voltage area constraints for specific nets, set the `ignore_voltage_areas` property on the nets by using the `set_zrt_net_properties` command. You can specify the nets either by using the `-nets` option to specify the nets on the command line or by specifying the nets in a file and using the `-from_file` option.

The following example shows the use of the `-nets` option:

```
icc_shell> set_zrt_net_properties -nets [get_nets my_net] \
-ignore_voltage_areas true
```

To get the value of the `ignore_voltage_areas` property on a net, use the `get_zrt_net_properties` command.

```
icc_shell> get_zrt_net_properties -net [get_nets my_net] \
    -property ignore_voltage_areas
```

When voltage area constraints are considered during routing, you can provide routing guidance for disjoint voltage areas by setting the `-voltage_area_weight` common route option to assign weights to each voltage area. Use the following syntax to assign weights to voltage areas:

```
set_route_zrt_common_options -voltage_area_weight {{VA_name weight} ...}
```

By default, all voltage areas have a weight of `medium`. To increase the cost of routing in a non-native voltage area, set the weight for that voltage area to `high`. To decrease the cost of routing in a non-native voltage area, set the weight for that voltage area to `low`.

To provide more detailed routing guidance for multivoltage designs, define voltage area routing rules by using the `define_voltage_area_routing_rule` command. A voltage area routing rule defines the set of voltage areas through which specific nets can pass. You can define one or more voltage area routing rules for a design. When routing a net that connects two voltage areas, the router evaluates the voltage area routing rules and allows nets that match the rules to pass through the specified voltage areas. These rules are honored by the virtual router and the global router; however, they are not honored during design planning hierarchical routing.

Note:

When voltage area routing rules are defined for a design, Zroute ignores the settings of the `-enforce_voltage_areas` and `-voltage_area_weight` common route options, as well as the `ignore_voltage_areas` net property.

In addition, the `define_voltage_area_routing_rule` command is mutually exclusive with the `set_fp_voltage_area_constraints` command; you should constrain voltage area routing using either one or the other.

By default, a voltage area routing rule applies to all nets. You can restrict the set of nets to those nets that connect certain voltage areas. To specify the connected voltage areas, use the `-to` and `-from` options to specify the voltage areas by using voltage area names, macro names, macro pins, or macro pads.

For example, to define a voltage area routing rule that allows only nets that connect the M1 cell and the VA1 voltage area to pass through the VA2 voltage area, use the following command:

```
icc_shell> define_voltage_area_routing_rule -rule va_rule1 \
    -from [get_cells M1] -to [get_voltage_area VA1] \
    -allow_pass_through VA2
```

By default, voltage area routing rules assign a medium cost to routing through the specified voltage areas. To change the cost associated with the voltage area routing rule, use the

`-pass_through_cost` option. To increase the cost of routing through the specified voltage areas, set the pass-through cost to `high`. To decrease the cost of routing through the specified voltage areas, set the pass-through cost to `low`.

For example, to discourage the routing of all nets through the voltage area named VA1, use the following command:

```
icc_shell> define_voltage_area_routing_rule -rule va_rule1 \
           -allow_pass_through {VA1} -pass_through_cost high
```

To report information about one or more voltage area routing rules, use the `report_voltage_area_routing_rule` command.

To remove one or more voltage area routing rules, use the `remove_voltage_area_routing_rule` command.

When you save the design in Milkyway format, the common route option settings, the net property settings, and the voltage area routing rules are saved with the design. For more information about setting common route options, see “[Setting Common Route Options](#)” on page 6-44.

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.

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 the 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. If the design contains existing global routes, by default, these global routes are ignored. To perform incremental global routing by reusing existing global routes, use the `-reuse_existing_global_route true` option. If the design contains existing detail routes for the specified nets, the `route_zrt_group` command performs incremental detail routing. 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 routing of clock nets:

- Balanced

To use balanced 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_options -routed_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 routing, use the `set_route_zrt_common_options` command to set the `-clock_topology` common route option to `comb` before routing the clock nets.

```
icc_shell> set_delay_calculation_options -routed_clock arnoldi
icc_shell> clock_opt -only_cts -no_clock_route
icc_shell> set_route_zrt_common_options -clock_topology comb
icc_shell> route_zrt_group -all_clock_nets
```

When comb routing is enabled, Zroute reports a DRC violation when a clock pin within the comb distance is not connected directly to the clock strap. These errors are saved in the error view, similar to other routing DRC violations.

- Normal

To use normal routing, use the `set_route_zrt_common_options` command to set the `-clock_topology` common route option to `normal` before routing the clock nets. Note that this is the default setting for this option.

```
icc_shell> set_delay_calculation_options -routed_clock arnoldi
icc_shell> clock_opt -only_cts -no_clock_route
icc_shell> set_route_zrt_common_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. For information about inserting redundant vias during clock routing, see “[Inserting Redundant Vias on Clock Nets](#)” on page 7-25. For information about postroute redundant via insertion, see “[Postroute Redundant Via Insertion](#)” on page 7-29.

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 6-16.

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 7-41.

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_sizing -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 Critical Nets

You can route a group of critical nets before routing the rest of the nets in the design by using the `route_zrt_group` command (or by choosing Route > Net Group Route in the GUI). You can specify the nets to route either by using the `-nets` option to specify the nets on the command line or by specifying the nets in a file and using the `-from_file` option:

```
icc_shell> route_zrt_group -nets collection_of_critical_nets
```

or

```
icc_shell> route_zrt_group -from_file file_name
```

If the specified nets are associated with a routing corridor, the nets are routed within the defined region. Note that global routing considers routing corridors as a hard constraint,

while track assignment and detail routing consider routing corridors as a soft constraint and might route nets slightly outside of the routing corridor to fix DRC violations.

The tool determines the order in which to route the specified nets and, by default, does not consider whether a net has a nondefault routing rule when determining the order. To route the nets with nondefault routing rules before the nets that use a default routing rule, set the `-route_nondefault_nets_first` option to `true`.

By default, the `route_zrt_group` command ignores existing global routes, reuses dangling wires and existing detail routes on the specified nets, and runs global routing, track assignment, and detail routing on the specified nets.

- To perform incremental global routing, set the `-reuse_existing_global_route` option to `true`.

Note:

You cannot use the `-reuse_existing_global_route true` option when routing the nets in a routing corridor. If you use this option, the global router ignores the routing corridors.

- To disable the reuse of dangling wires, set the `-utilize_dangling_wires` option to `false`.
- To stop after global routing, set the `-stop_after_global_route` option to `true`.

During detail routing, Zroute also performs search and repair. By default, the detail router performs a maximum of 40 iterations. If Zroute determines before then that all violations have been fixed or that it cannot fix the remaining violations, it stops. You can control the maximum number of detail routing iterations by using the `-max_detail_route_iterations` option.

By default, the `route_zrt_group` command does not fix soft DRC violations, such as bridge rule violations. To enable the fixing of soft DRC violations after the final detail routing iteration, use the `set_route_zrt_common_options` command to set the `-post_group_route_fix_soft_violations` common route option to `true`.

```
icc_shell> set_route_zrt_common_options \
-post_group_route_fix_soft_violations true
```

Routing Secondary Power and Ground Pins

To use Zroute to perform secondary power and ground pin routing,

1. Verify that the secondary power and ground pins have the appropriate attributes in the Milkyway library (FRAM view).
2. Set the routing constraints for secondary power and ground pin routing.
3. Perform secondary power and ground pin routing by using the `route_zrt_group` command.

The following sections describe these steps.

Note:

If the standard cells in your design use secondary power and ground rails rather than secondary power and ground pins, use the `preroute_standard_cells -mode rail` command to connect these rails.

Verifying the Secondary Power and Ground Pin Attributes

The IC Compiler tool supports four types of secondary power and ground pins:

- Internal power pin
- Internal ground pin
- Backup power pin
- Backup ground pin

These secondary power and ground pins are identified by the `pg_type` attribute in the logic (.db) library and the `PORT_TYPE` attribute in the FRAM view of the Milkyway library. [Table 6-6](#) shows the correlation between the secondary power and ground pin types, the `pg_type` attributes, and the `PORT_TYPE` attributes.

Table 6-6 Attributes for Secondary Power and Ground Pins

Secondary power or ground pin type	<code>pg_type</code> attribute in logic (.db) library	<code>PORT_TYPE</code> attribute in Milkyway FRAM view
Internal power pin	<code>internal_power</code>	"Power" "InternalPG"
Internal ground pin	<code>internal_ground</code>	"Ground" "InternalPG"

Table 6-6 Attributes for Secondary Power and Ground Pins (Continued)

Secondary power or ground pin type	pg_type attribute in logic (.db) library	PORT_TYPE attribute in Milkyway FRAM view
Backup power pin	backup_power	"Power" "SecondaryPG"
Backup ground pin	backup_ground	"Ground" "SecondaryPG"

Before you use Zroute to perform secondary power and ground pin routing, you must verify that the libraries have the correct attributes on the secondary power and ground pins of the standard cells. For details about verifying and updating the libraries, see Appendix A, “Updating Secondary Power and Ground Pins Application Note,” in the *UPF Library Preparation Application Notes*.

Setting the Routing Constraints

You can set routing constraints for secondary power and ground pin routing in the same way as for regular signal routing. For example, you can set constraints by

- Defining the minimum and maximum routing layers by using the `set_net_routing_layer_constraints` command
For more information about using the `set_net_routing_layer_constraints` command, see “[Specifying Net-Specific Routing Layer Constraints](#)” on page 6-15.
- Specifying the preferred pin connections by using the `set_route_zrt_common_options` command to set the `-single_connection_to_pins` and `-connect_within_pins_by_layer_name` common route options

For example, to require a single connection to the secondary power and ground pins and require that the metal1 connections use vias contained within the pin shapes, use the following command:

```
icc_shell> set_route_zrt_common_options \
-single_connection_to_pins standard_cell_pins \
-connect_within_pins_by_layer_name {metal1 via_standard_cell_pins}
```

- Defining the maximum number of power or ground pins in a cluster by using the `set_route_zrt_common_options` command to set the `-number_of_secondary_pg_pin_connections` common route option

A cluster is a set of connected secondary power or ground pins that has one connection to a PG strap or ring. By default, the value of this option is 0, which means that there is no limit on the number of secondary power or ground pins in a cluster.

For example, to connect all secondary power and ground pins directly to a PG strap or ring, use the following command:

```
icc_shell> set_route_zrt_common_options \
    -number_of_secondary_pg_pin_connections 1
```

- Defining a nondefault routing rule for secondary power and ground pin routing

For example, to define a nondefault routing rule named wideSVDD for wide metal1 and metal2 and set the nondefault routing rule on the VDD2 net, to which the secondary power and ground pins are connected, use the following commands:

```
icc_shell> define_routing_rule wideSVDD -widths { M1 0.3 M2 0.3 }
icc_shell> set_net_routing_rule -rule wideSVDD {VDD2}
```

For more information about using nondefault routing rules, see “[Using Nondefault Routing Rules](#)” on page 6-16.

By default, the constraints apply to both the secondary power and ground connections and the tie-off connections. To separate these connections so that you can set constraints only for the secondary power and ground connections, use the `set_route_zrt_common_options` command to set the `-separate_tie_off_from_secondary_pg` common route option to `true`.

Routing the Secondary Power and Ground Pins

If the secondary power and ground pins have the appropriate attributes in the Milkyway FRAM view, you can use Zroute to route the secondary power and ground pins.

For example, to connect the secondary power pins to the VDD1 net, run the following command:

```
icc_shell> route_zrt_group -nets VDD1
```

The following example add new cells, which have secondary power and ground pins, to a design; logically connects the power and ground pins; and then connects the secondary power pins to the VDD2 net.

```
icc_shell> insert_buffer {TOP/U1001/Z} {libA/BUFHVT} \
    -new_cell_names mynewHVT -location {25.0 50.0}
icc_shell> insert_buffer {TOP/U1002/Z} {libA/BUFHVT} \
    -new_cell_names mynewHVT -location {25.0 100.0}
icc_shell> legalize_placement -incremental
icc_shell> derive_pg_connection
icc_shell> route_zrt_group -nets {VDD2}
```

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.

If you need to customize your routing flow or you need to run a large design step-by-step, you might want to use the standalone routing commands instead of the `route_opt` command or automatic routing.

- 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 Signal Nets” on page 7-26](#).

Routing Signal Nets by Using Individual Routing Commands

After you have specified your routing options, you can debug your design or monitor each routing step by performing each routing step individually. Individual routing steps are explained in the following sections:

- [Global Routing](#)
- [Track Assignment](#)
- [Detail Routing](#)

Global Routing

Before you run global routing, you must define the Zroute common route options, as described in [“Setting Common Route Options” on page 6-44](#), and the Zroute global route options, as described in [“Setting Global Route Options” on page 6-54](#).

To perform standalone global routing, use the `route_zrt_global` command (or choose Route > Global Route in the GUI).

The global router divides a design into global routing cells. By default, the width of a global routing cell is the same as the height of a standard cell and is aligned with the standard cell rows.

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. The tool calculates the demand for wire tracks in each global routing cell and reports the overflows, which are the number of wire tracks that are still needed after the tool assigns nets to the available wire tracks in a global routing cell.

Global routing is done in two phases:

- The initial routing phase (phase 0), in which the tool routes the unconnected nets and calculates the overflow for each global routing cell
- The rerouting phases, in which the tool tries to reduce congestion by ripping up and rerouting nets around global routing cells with overflows

The tool might perform several rerouting phases. At the end of each rerouting phase, the tool recalculates the overflows. You should see a reduction in the total number of global routing cells with overflow and in the total overflow numbers. The global router stops and exits from the rerouting phase when the congestion is solved or cannot be solved further or after the maximum number of phases has occurred, as defined by the `-effort` option. You can force the global router to perform the maximum number of phases based on the specified effort level by using the `set_route_zrt_global_options` command to set the `-force_full_effort` global route option to `true`. By default, the tool uses medium

effort and performs a maximum of three rerouting phases. You can perform up to six rerouting phases by specifying ultra effort.

There are five global routing effort levels: `minimum`, `low`, `medium`, `high`, and `ultra`.

- **Minimum** (`-effort minimum`)

The minimum effort level uses two times larger global routing cells relative to the other effort levels. It also has a much lower congestion cost and runs only one rerouting phase. It should only be used for prototype routing or for an initial congestion evaluation, not for detail routing.

- **Low** (`-effort low`)

Low effort runs a maximum of two rerouting phases with very similar congestion cost. It is faster in comparison to medium effort and has reasonable QoR. If your design is not very congested, you can use the low effort level.

- **Medium** (`-effort medium`)

Medium effort is the default effort level and runs a maximum of three rerouting phases. Global routing stops after the third phase or when the overflow is resolved, whichever occurs first.

- **High** (`-effort high`)

High effort runs up to four rerouting phases. If your design is congested, use the high effort level.

- **Ultra** (`-effort ultra`)

Ultra effort runs up to six rerouting phases. If your design is very congested, use the ultra effort level.

At the end of global routing, the following information is stored in the Milkyway design database:

- The g-links and g-vias on each routed net

This information is used for the next routing steps. After Zroute performs track assignment and detail routing, it removes these g-links and g-vias from the design database.

- The congestion data

By default, the congestion data for each layer is stored in the Milkyway design database, which enables the tool to generate a layer-based congestion map; however, saving the layer-based information increases the size of the Milkyway design database. If you do not need layer-based congestion maps, you can disable this capability and save only aggregate congestion information by using the `set_route_zrt_global_options` command to set the `-layer_based_congestion_map` global route option to `false` before running global routing.

The global router reports design statistics and congestion data after the initial routing phase and after each rerouting phase. When global routing is complete, the global router reports a summary of the wire length and via count.

[Example 6-2](#) shows a global routing report. In the congestion report, the Overflow value is the total number of wires in the design that do not have a corresponding track available. The Max value corresponds to the highest number of overutilized wires in a single global routing cell. The GRCs value is the total number of overcongested global routing cells in the design.

Example 6-2 Global Routing Report

```

Start Global Route ...
[Init] Elapsed real time: 0:00:00
[Init] Elapsed cpu  time: sys=0:00:00 usr=0:00:00 total=0:00:00
.....
Begin global routing.
Constructing data structure ...
Design statistics:
Design Bounding Box (0.00,0.00,3180.00,1154.00)
Number of routing layers = 10
layer M1, dir Ver, min width = 0.09, min space = 0.09 pitch = 0.24
layer M2, dir Hor, min width = 0.10, min space = 0.10 pitch = 0.20
layer M3, dir Ver, min width = 0.10, min space = 0.10 pitch = 0.30
layer M4, dir Hor, min width = 0.10, min space = 0.10 pitch = 0.30
layer M5, dir Ver, min width = 0.10, min space = 0.10 pitch = 0.30
layer M6, dir Hor, min width = 0.10, min space = 0.10 pitch = 0.30
layer M7, dir Ver, min width = 0.10, min space = 0.10 pitch = 0.30
...
Net statistics:
Total number of nets      = 231242
Number of nets to route   = 231177
Number of single or zero port nets = 64
1 nets are partially routed.
1 nets are partially drouuted.
0 nets are partially grouted.
1 nets are routed.
1 nets are drouuted.
0 nets are grouted.
4 nets have non-default rule non_m2_route_rules_for_noise
12 nets have non-default rule clockRouteRuleCTS
...
phase3. Routing result:
phase3. Both Dirs: Overflow = 453 Max = 4 GRCs =    449 (0.02%)
phase3. H routing: Overflow = 148 Max = 1 (GRCs = 114) GRCs = 202 (0.02%)
phase3. V routing: Overflow = 304 Max = 4 (GRCs =  2) GRCs = 247 (0.03%)
phase3. M1          Overflow =  1 Max = 0 (GRCs =  2) GRCs =    2 (0.00%)
phase3. M2          Overflow = 34 Max = 0 (GRCs = 88) GRCs =   88 (0.01%)
phase3. M3          Overflow = 184 Max = 4 (GRCs =  2) GRCs = 126 (0.01%)
phase3. M4          Overflow =  80 Max = 1 (GRCs = 80) GRCs =   80 (0.01%)
phase3. M5          Overflow =  0 Max = 0 (GRCs =  0) GRCs =     0 (0.00%)
phase3. M6          Overflow = 34 Max = 1 (GRCs = 34) GRCs =   34 (0.00%)
phase3. M7          Overflow = 119 Max = 1 (GRCs = 119) GRCs = 119 (0.01%)

```

```
.....phase3. Total Wire Length = 16820202.00
phase3. Layer M1 wire length = 10412.66
phase3. Layer M2 wire length = 3576314.50
phase3. Layer M3 wire length = 2721198.75
phase3. Layer M4 wire length = 2988112.75
phase3. Layer M5 wire length = 2201333.00
phase3. Layer M6 wire length = 4374871.00
phase3. Layer M7 wire length = 947959.25
phase3. Layer M8 wire length = 0.00
phase3. Layer M9 wire length = 0.00
phase3. Layer M10 wire length = 0.00
phase3. Total Number of Contacts = 1820147
phase3. Via VIA12 count = 749084
phase3. Via VIA23 count = 762968
phase3. Via VIA34 count = 142823
phase3. Via VIA45 count = 91356
phase3. Via VIA56 count = 46191
phase3. Via VIA67 count = 27725
phase3. Via VIA78 count = 0
phase3. Via VIA89 count = 0
phase3. Via VIA910mH count = 0
phase3. completed.
```

Before proceeding to detail routing, display the congestion map in the GUI, and check the overflow distribution. The congestion report and map help you to identify congested areas. For more information about the congestion report and map, see “[Analyzing Congestion](#)” on page [6-114](#).

Global Routing During Design Planning

During design planning you can perform exploration-mode global routing by using the `route_zrt_global -exploration true` command.

```
icc_shell> route_zrt_global -exploration true
```

If you are using the hierarchical flow, enable plan-group-aware global routing by using the `set_route_zrt_common_options` command to set the `-plan_group_aware` common route option before running the `route_zrt_global` command. When you set the `-plan_group_aware` option to `all_routing`, Zroute routes all the nets in the design and preserves the hierarchy and pin constraints. You can increase the plan-group-aware routing speed by routing only the top-level nets. To do this, set the `-plan_group_aware` option to `top_level_routing_only`. When the `-plan_group_aware` option is set to `off`, which is the default, Zroute ignores the plan groups and routes the design as flat.

```
icc_shell> set_route_zrt_common_options -plan_group_aware all_routing
icc_shell> route_zrt_global -exploration true
```

Timing-Driven Global Routing

By default, the `route_zrt_global` command is not timing-driven. To enable timing-driven global routing, use the `set_route_zrt_global_options` command to set the `-timing_driven` global route option before you run the `route_zrt_global` command.

By default, when you enable timing-driving global routing, Zroute uses medium effort to assign timing-critical nets to low-resistance metal layers. To change the extent to which timing-driven global routing prefers lower-resistance metal layers when routing timing-critical nets, use the `set_route_zrt_common_options` command to set the `-rc_driven_setup_effort_level` common route option. To increase the effort level to use more low-resistance metal layers for routing, set the option to `high`. To reduce the effort level, set the option to `low`. To disable resistance-based routing layer preferences, set the option to `off`.

You can control the tradeoff between timing QoR and DRC convergence by using the `set_route_zrt_global_options` command to set the `-timing_driven_effort_level` global route option. By default, this option has a setting of `high`, which favors timing QoR over DRC convergence.

When you enable timing-driven global routing, the tool calculates the net delays before invoking the global router. If the design database contains global route information, the tool uses the global route information to calculate the net delays; otherwise, it uses virtual routing to calculate the net delays. The global routing results can vary depending on whether the initial net delays were calculated by using global route information or virtual routing. To remove existing global route information from the signal nets in the design, use the `-signal_global_route` option with the `remove_route_by_type` command, as shown in the following example:

```
icc_shell> remove_route_by_type -signal_global_route
```

Crosstalk-Driven Global Routing

By default, the `route_zrt_global` command is not crosstalk-driven. To enable crosstalk-driven global routing, set the `route_xtalk_prevention` signal integrity option, as described in [“Setting Crosstalk Prevention Options” on page 6-65](#).

When you enable crosstalk-driven global routing, the tool calculates the net delays before invoking the global router. If the design database contains global route information, the tool uses the global route information to calculate the net delays; otherwise, it uses virtual routing to calculate the net delays. The global routing results can vary depending on whether the initial net delays were calculated by using global route information or virtual routing. To remove existing global route information from the signal nets in the design, use the `-signal_global_route` option with the `remove_route_by_type` command, as shown in the following example:

```
icc_shell> remove_route_by_type -signal_global_route
```

Incremental Global Routing

By default, the global router ignores existing global routes. To perform incremental global routing by reusing the existing global routes, use the `-reuse_existing_global_route true` option when you run global routing. Note that this option affects only the global router and not the net delay calculation that occurs before timing-driven or crosstalk-driven global routing.

Track Assignment

Before you run track assignment, you must define the Zroute common route options, as described in “[Setting Common Route Options](#)” on page 6-44, and the Zroute track assignment options, as described in “[Setting Track Assignment Options](#)” on page 6-57.

To perform standalone track assignment, run the `route_zrt_track` command (or choose Route > Track Assignment in the GUI).

The main task of track assignment is to assign routing tracks for each global route. During track assignment, Zroute performs the following tasks:

- Assigns tracks in horizontal partitions.
- Assigns tracks in vertical partitions.
- Reroutes overlapping wires.

After track assignment finishes, all nets are routed but not very carefully. There are many violations, particularly where the routing connects to pins. Detail routing works to correct those violations.

Note:

Because track assignment replaces the global routes with actual metal layers, the design no longer contains global routes after track assignment completes.

By default, the `route_zrt_track` command is not timing-driven or crosstalk-driven.

- To enable timing-driven mode, set the `timing_driven` track assignment option.
- To enable crosstalk-driven mode, set the `route_xtalk_prevention` signal integrity option, as described in “[Setting Crosstalk Prevention Options](#)” on page 6-65.

At the end of track assignment, Zroute reports a summary of the wire length and via count. [Example 6-3](#) shows a track assignment report.

Example 6-3 Track Assignment Report

Wire length and via report:

```
-----
Number of metal1 wires: 90644          via01: 0
Number of metal2 wires: 1305119        via12: 1171053
Number of metal3 wires: 823570         via23: 1300628
Number of metal4 wires: 124268         via34: 192435
Number of metal5 wires: 28001          via45: 44130
Number of metal6 wires: 4054           via56: 7644
Number of metal7 wires: 0              via67: 0
Total number of wires: 2375656        vias: 2715890

Total metal1 wire length: 170860.9
Total metal2 wire length: 2493004.5
Total metal3 wire length: 4127489.2
Total metal4 wire length: 2109298.0
Total metal5 wire length: 1584743.5
Total metal6 wire length: 300718.1
Total metal7 wire length: 0.0
Total wire length: 10786114.0

Longest metal1 wire length: 540.7
Longest metal2 wire length: 603.5
Longest metal3 wire length: 729.7
Longest metal4 wire length: 821.1
Longest metal5 wire length: 958.2
Longest metal6 wire length: 935.7
Longest metal7 wire length: 0.0
```

After track assignment, you can display a congestion report and map that are based on the track assignment results. For more information about the congestion report and map, see [“Analyzing Congestion” on page 6-114](#).

Detail Routing

Before you run detail routing, you must define the Zroute common route options, as described in [“Setting Common Route Options” on page 6-44](#), and the Zroute detail route options, as described in [“Setting Detail Route Options” on page 6-57](#).

The detail router uses the general pathways suggested by global routing and track assignment to route the nets, and then it divides the design into partitions and looks for DRC violations in each partition. When the detail router finds a violation, it rips up the wire and reroutes it to fix the violation. During detail routing, Zroute concurrently addresses routing design rules and antenna rules and optimizes via count and wire length. For more information about antenna rules and via count and wire length optimization, see [Chapter 7, “Chip Finishing and Design for Manufacturing.”](#)

To perform standalone detail routing, run the `route_zrt_detail` command (or choose Route > Detail Route in the GUI).

By default, the `route_zrt_detail` command

- Performs detail routing on the whole design

You can restrict the routing to a specific area of the design by using the `-coordinates` option (or by specifying or selecting the bounding box in the GUI).

- Uses one uniform partition for the first iteration and adjusts the partitioning for subsequent iterations

Zroute uses the single uniform partition for the first iteration to generate all DRC violations for the chip at the same time. At the beginning of each subsequent iteration, the router checks the distribution of the DRC violations. If the DRC violations are evenly distributed, the detail router uses a uniform partition. If the DRC violations are located in some local areas, the detail router uses nonuniform partitions.

- Performs iterations until one of the following conditions exists:

- All of the violations have been fixed
- The maximum number of iterations has been reached

By default, the maximum number of iterations is 40. You can change this limit by setting the `-max_number_iterations` option.

```
icc_shell> route_zrt_detail -max_number_iterations 20
```

- It cannot fix any of the remaining violations

You can change the effort that the detail router uses for fixing the remaining violations before it gives up by using the `set_route_zrt_detail_options` command to set the `-drc_convergence_effort_level` detail route option.

```
icc_shell> set_route_zrt_detail_options \  
-drc_convergence_effort_level high
```

You can force the detail router to complete the maximum number of iterations, regardless of the DRC convergence status, by using the `set_route_zrt_detail_options` command to set the `-force_max_number_iterations` detail route option to `true`.

```
icc_shell> set_route_zrt_detail_options \  
-force_max_number_iterations true
```

- Is not timing-driven

To enable timing-driven detail routing, use the `set_route_zrt_detail_options` command to set the `-timing_driven` detail route option.

```
icc_shell> set_route_zrt_detail_options -timing_driven true
```

Note:

If you perform detail routing by using the `route_opt` command, the default behavior is timing-driven detail routing.

By default, when you enable timing-driving detail routing, Zroute uses medium effort to assign timing-critical nets to low-resistance metal layers. To change the extent to which timing-driven detail routing prefers lower-resistance metal layers when routing timing-critical nets, use the `set_route_zrt_common_options` command to set the `-rc_driven_setup_effort_level` common route option. To increase the effort level to use more low-resistance metal layers for routing, set the option to `high`. To reduce the effort level, set the option to `low`. To disable resistance-based routing layer preferences, set the option to `off`.

- Does not fix shorted nets over macro cells

If default detail routing leaves shorted nets over macro cells, analyze the design to determine if the shorts are caused by the availability of only a single layer for routing over the macro cells. If so, enable river routing by creating single-layer route guides over the macros with shorted nets and rerun detail routing. For information about creating route guides, see “[Defining Route Guides](#)” on page 6-28.

If shorted nets remain after using river routing, enable the fixing of shorted nets over macro cells by automatically ripping up and rerouting the shorted nets by using the `set_route_zrt_detail_options` command to set the `-repair_shorts_over_macros_effort_level` detail route option to `low`, `medium`, or `high` and running incremental detail routing. The higher the effort level, the more ECO routing iterations are performed, which can reduce the number of DRC violations at the expense of runtime.

```
icc_shell> set_route_zrt_detail_options \
-repair_shorts_over_macros_effort_level high
```

- Does not fix soft DRC violations, such as bridge rule violations

To enable the fixing of soft DRC violations after the final detail routing iteration, use the `set_route_zrt_common_options` command to set the `-post_detail_route_fix_soft_violations` common route option to `true`.

```
icc_shell> set_route_zrt_common_options \
-post_detail_route_fix_soft_violations true
```

You can run additional detail routing iterations on a routed design by running incremental detail routing (the `-incremental` option). Be sure to use the `-incremental` option; otherwise, Zroute restarts at iteration 0 with a fixed-size partition.

```
icc_shell> route_zrt_detail -incremental true
```

By default, incremental detail routing does not fix soft DRC violations, such as bridge rule violations. To enable the fixing of soft DRC violations after the final incremental detail routing iteration, use the `set_route_zrt_common_options` command to set the

-post_incremental_detail_route_fix_soft_violations common route option to true.

```
icc_shell> set_route_zrt_common_options \
           -post_incremental_detail_route_fix_soft_violations true
```

Note:

Incremental detail routing does not fix open nets. To fix open nets, you must run ECO routing. For information about ECO routing, see “[Performing ECO Routing](#)” on [page 6-110](#).

If you want to view the DRC violations before postroute optimization, you can save the design after a specified number of iterations by using the `set_route_zrt_detail_options` command to set the `-save_after_iterations` detail route option. The saved design is called `DR_itrn`, where `n` is the specified iteration. You can use a string other than `DR` as the prefix by using the `set_route_zrt_detail_options` command to set the `-save_cell_prefix` detail route option.

Zroute generates a DRC violations summary at the end of each iteration. For more information about the DRC violations reported by Zroute, see “[Performing Design Rule Checking Using the IC Compiler Tool](#)” on [page 6-120](#).

After completing detail routing, Zroute outputs a final summary report in which the redundant violations are merged. This report includes all violations detected by Zroute. If you want an additional report that excludes violations that are not of interest to you, specify the rules to exclude by using the `set_route_zrt_detail_options` command to set the `-report_ignore_drc` detail route option. The syntax to set this option is

```
set_route_zrt_detail_options -report_ignore_drc list_of_drcs
```

The values used in the `list_of_drcs` argument are the DRC names used in the summary report. If the DRC name includes a space, you must enclose the name in double quotation marks. For a complete list of the supported DRC names, see the man page.

[Example 6-4](#) shows a detail routing report.

Example 6-4 Detail Routing Report

```

Start DR iteration 0: uniform partition
Routed 1/5734 SBoxes, Violations = 0
Routed 28/5734 SBoxes, Violations = 14
...
DRC-SUMMARY:
    @@@@@ TOTAL VIOLATIONS = 437
    @@@ Total number of instance ports with antenna violations = 690
        Diff net spacing : 300
            End of line spacing : 1
            Same net spacing : 6
            Same net via-cut spacing : 2
            Less than minimum area : 20
            Less than minimum edge length : 19
            Needs fat contact : 5
            Needs fat contact on extension : 7
            Internal-only types : 77

        End DR iteration 21 with 198 parts
        @@@@ Total nets not meeting constraints = 1

Stop DR since not converging
...
Information: Merged away 13 aligned/redundant DRCs. (RT-805)

DR finished with 47 violations and 144 instance ports antenna violations

        Diff net spacing : 31
        Less than minimum edge length : 1
        Internal-only types : 15

Total Wire Length = 11107954 micron
Total Number of Contacts = 2787466
Total Number of Wires = 2759250
Total Number of PtConns = 598905
Total Number of Non-Default Contacts = 362
    Layer      M1 : 190733 micron
    Layer      M2 : 2474471 micron
    Layer      M3 : 4106791 micron
    Layer      M4 : 2377087 micron
    Layer      M5 : 1701419 micron
    Layer      M6 : 257452 micron
    Layer      M7 : 0 micron
    Via       VIA56 : 9778
    Via       VIA45 : 76055

```

After detail routing, you can display a congestion report and map that are based on the detail routing results. For more information about the congestion report and map, see “[Analyzing Congestion](#)” on page 6-114.

Routing Signal Nets by Using Automatic Routing

Before you use automatic routing to route the signal nets, you must define the Zroute routing options (common route, global route, track assignment, and detail route), as described in “[Setting Zroute Options](#)” on page 6-43.

To run automatic routing, use the `route_zrt_auto` command (or choose Route > Auto Route in the GUI). By default, the `route_zrt_auto` command ignores existing global routes and sequentially invokes global routing, track assignment, and detail routing. You can perform incremental global routing by using the `-reuse_existing_global_route true` option. You can stop after track assignment by using the `-stop_after_track_assignment` option.

By default, the `route_zrt_auto` command is not timing-driven or crosstalk-driven.

- To enable timing-driven mode, set the `timing_driven` global route, track assignment, and detail route options.

By default, when you enable timing-driving routing, Zroute uses medium effort to assign timing-critical nets to low-resistance metal layers. To change the extent to which timing-driven routing prefers lower-resistance metal layers when routing timing-critical nets, use the `set_route_zrt_common_options` command to set the `-rc_driven_setup_effort_level` common route option. To increase the effort level to use more low-resistance metal layers for routing, set the option to `high`. To reduce the effort level, set the option to `low`. To disable resistance-based routing layer preferences, set the option to `off`.

- To enable crosstalk-driven mode for global routing and track assignment, set the `route_xtalk_prevention` signal integrity option, as described in “[Setting Crosstalk Prevention Options](#)” on page 6-65.

By default, the `route_zrt_auto` command does not fix soft DRC violations, such as bridge rule violations. To enable the fixing of soft DRC violations after the final detail routing iteration, use the `set_route_zrt_common_options` command to set the `-post_detail_route_fix_soft_violations` common route option to `true`.

```
icc_shell> set_route_zrt_common_options \
-post_detail_route_fix_soft_violations true
```

[Table 6-7](#) describes the `route_zrt_auto` options.

Table 6-7 route_zrt_auto Command Options

Command option	Description
<code>-stop_after_track_assignment true false</code> ("Stop after track assignment" check box in the GUI)	Stops after performing track assignment. The default is <code>false</code> .
<code>-save_after_global_route true false</code> ("Save cell after: Global route" check box in the GUI)	Controls whether the design is saved after global routing. When <code>true</code> , the tool saves the design in a CEL view named <code>auto_GR</code> . The default is <code>false</code> .
<code>-save_after_track_assignment true false</code> ("Save cell after: Track assignment" check box in the GUI)	Controls whether the design is saved after track assignment. When <code>true</code> , the tool saves the design in a CEL view named <code>auto_TA</code> . The default is <code>false</code> .
<code>-save_after_detail_route true false</code> ("Save cell after: Detail route" check box in the GUI)	Controls whether the design is saved after detail routing. When <code>true</code> , the tool saves the design in a CEL view named <code>auto_DR</code> . The default is <code>false</code> .
<code>-max_detail_route_iterations cnt</code> ("Maximum detail router iterations" box in the GUI)	The maximum number of detail routing iterations. The default is the 40.
<code>-save_cell_prefix string</code> ("Save cell prefix" box in the GUI)	Specifies the prefix to use for intermediate saved designs. The default is <code>auto</code> .
<code>-reuse_existing_global_route true false</code> ("Reuse existing global route" check box in the GUI)	Performs incremental global routing. The default is <code>false</code> .

Routing Signal Nets by Using the `route_opt` Command

Before you use the `route_opt` command to route the signal nets, you must set the Zroute routing options (common route, global route, track assignment, and detail route), as described in “[Setting Zroute Options](#)” on page 6-43; set the signal integrity options as described in “[Setting Crosstalk Prevention Options](#)” on page 6-65; 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 6-104.

Setting the Routing and Optimization Strategy

The IC Compiler tool provides strategy controls that you set to guide how routing and postroute optimizations are run. These settings affect the `route_opt` and `focal_opt` commands.

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).

[Table 6-8](#) describes the `set_route_opt_strategy` options.

Table 6-8 set_route_opt_strategy Options

Command option	Description
<code>-fix_hold_mode all route_base</code> (“Fix hold optimization stage” area in the GUI)	The stages for which hold optimization is performed. You can select prerouting, global routing, and detail routing (<code>all</code>) or global routing and detail routing (<code>route_base</code>). The default is <code>route_base</code> .
<code>-power_aware_optimization true false</code> (“Power aware optimization flow” check box in the GUI)	Controls whether postroute optimizations for setup, hold, and design rule constraints consider power. The default is <code>false</code> .
<code>-enable_port_punching true false</code> (“Enable port punching” check box in the GUI)	Controls whether postroute optimizations can modify the logical topology of a net to increase the area available for buffer insertion to fix constraint violations. The default is <code>false</code> .

Table 6-8 set_route_opt_strategy Options (Continued)

Command option	Description
-fast_pba_analysis true false ("Enable fast path-based advanced on chip variation analysis" box in the GUI)	Controls whether postroute optimizations use the fast path-based analysis timing adjustment when AOCV is enabled. The default is <code>false</code> .
-xtalk_reduction_loops int ("Maximum crosstalk reduction cycles" box in the GUI)	The maximum number of crosstalk reduction iterations. The default is 1.
-search_repair_loops int ("Maximum initial route Search and Repair cycles" box in the GUI)	The maximum number of initial routing iterations. The Zroute default is 10 (the classic router default is 15).
-eco_route_search_repair_loops int ("Maximum ECO Search and Repair cycles" box in the GUI)	The maximum number of ECO iterations. Valid values are between -1 and 500. To disable ECO routing, set this option to -1. The Zroute default is 4 (the classic router default is 5).
-route_drc_threshold int ("Route violations threshold to trigger the reduction Search and Repair loop" box in the GUI)	The threshold number of routing violations before limiting detail routing to one iteration. The default is 3000.

To report your settings, use the `report_route_opt_strategy` command.

Using Scenario Compression for Multicorner-Multimode Designs

For multicorner-multimode designs that have fewer modes than scenarios, you can improve the capacity and runtime of postroute optimization without sacrificing QoR by enabling scenario compression. When you enable scenario compression, the original set of scenarios is compressed into a subset of dominant scenarios. In addition, the constraints of this dominant subset are modified to capture any violations from the nondominant scenarios. To identify the dominant scenarios used during compression, the tool appends a suffix of `_z_` to the scenario names.

To use scenario compression,

1. If your design is not yet routed, perform initial routing by running the `route_opt -initial_route_only` command.

```
icc_shell> route_opt -initial_route_only
```

2. Enable scenario compression by running the `compress_scenarios` command.

```
icc_shell> compress_scenarios
```

By default, the tool determines the number of dominant scenarios. To specify a maximum number of dominant scenarios, use the `-max_compression` option with the `compress_scenarios` command.

3. Run the `route_opt` command one or more times to perform initial or incremental postroute optimization.

Note:

For the best runtime benefits, run the `route_opt` command at least two times.

To run initial postroute optimization, use the `route_opt -skip_initial_route` command. To run incremental postroute optimization, use the `route_opt -incremental` command. You can use other `route_opt` options as necessary to configure the postroute optimization.

4. Disable scenario compression by running the `uncompress_scenarios` command.

```
icc_shell> uncompress_scenarios
```

Note:

Scenario compression is supported only by the `route_opt` command with the `-skip_initial_route` or `-incremental` option. You must revert your scenarios back to the uncompressed state before running any other commands, including timing analysis commands.

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, as described in [“Setting Crosstalk Reduction Options” on page 6-66](#).

- 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, as described in [“Setting Crosstalk Reduction Options” on page 6-66](#).

- Detail routing

By default, the `route_opt` command performs 10 detail routing iterations of timing-driven detail routing and does not do crosstalk reduction.

To change the number of detail routing iterations, use the `-search_repair_loops` option with the `set_route_opt_strategy` command to specify the number iterations. If Zroute detects more than 3000 routing DRC violations during the first iteration, it stops detail routing. To change this limit, use the `-route_drc_threshold` option with the `set_route_opt_strategy` command to specify the limit. To disable this check, set this option to -1.

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, enable signal integrity mode and use the `-xtalk_reduction` or `-only_xtalk_reduction` option when you run the `route_opt` command, as described in “[Setting Crosstalk Reduction Options](#)” on page 6-66.

By default, the `route_opt` command does not fix soft DRC violations, such as bridge rule violations. To enable the fixing of soft DRC violations after the final detail routing iteration, use the `set_route_zrt_common_options` command to set the `-post_detail_route_fix_soft_violations` common route option to true.

```
icc_shell> set_route_zrt_common_options \
           -post_detail_route_fix_soft_violations true
```

- Postroute optimization

By default, the `route_opt` command performs timing optimization to fix setup, hold, and logical design constraint violations. In addition, it can perform the following optimizations: leakage power, signal integrity, and area recovery. For detailed information about postroute optimization, see “[Performing Postroute Optimization](#)” on page 6-98.

For designs that contain block abstractions, the `route_opt` command performs top-level interface optimization during postroute optimization. For information about top-level interface optimization, see “[Performing Transparent Interface Optimization](#)” on page 10-13.

After each optimization loop, the `route_opt` command performs ECO routing to connect open nets and fix any DRC violations that occurred during optimization. By default, the command performs four ECO routing iterations. To change the number of ECO iterations, set the `-eco_route_search_repair_loops` option of the `set_route_opt_strategy` command. To disable the ECO routing, set this option to -1. If you disable ECO routing by the `route_opt` command, you must perform standalone ECO routing to connect the open nets and fix any DRC violations.

Note:

Although the `route_opt` command performs timing-driven routing by default, the Zroute basic routing commands do not. You must explicitly enable timing-driven routing for the basic routing commands by setting the appropriate `timing_driven` Zroute routing option.

If you want to analyze the routing results before performing postroute optimization or you want to use scenario compression during postroute optimization, first run the `route_opt` command without postroute optimization (`-initial_route_only` option), and then run the `route_opt` command with only postroute optimization.

For example, to analyze the routing results before performing postroute optimization, use the following flow:

```
icc_shell> route_opt -initial_route_only
# analyze routing results
icc_shell> route_opt -skip_initial_route
```

To use scenario compression during postroute optimization, use the following flow:

```
icc_shell> route_opt -initial_route_only
icc_shell> compress_scenarios
icc_shell> route_opt -skip_initial_route
icc_shell> route_opt -incremental
icc_shell> uncompress_scenarios
```

For more information about using scenario compression, see “[Using Scenario Compression for Multicorner-Multimode Designs](#)” on page 6-95.

Saving the Intermediate Results

By default, when you run the `route_opt` command with Zroute, the tool automatically saves the design after two iterations (iteration 1) to allow you to view the DRC violations before postroute optimization. The saved design is called `cell_name_INIT_RT_itr1`. You can select an iteration other than iteration 1 by using the `-save_after_iterations` option. You can use a string other than the cell name as the prefix by using the `-save_cell_prefix` option. These options are supported only when using the `route_opt` command with Zroute.

Note:

If you run the `route_opt -initial_route_only` command, the tool does not automatically save the design after two iterations. However, you can force the tool to save the design by using the `-save_after_iterations` option.

In addition, Zroute supports the standard `route_opt` checkpointing. To enable `route_opt` checkpointing, set the `routeopt_checkpoint` variable to `true`.

If you also use the `-save_after_iterations` and `-save_cell_prefix` options, they are honored only for the first version of the design saved during checkpointing. They are not honored for subsequent checkpoint saves.

Performing Postroute Optimization

The IC Compiler tool supports two types of postroute optimization:

- Standard postroute optimization, which always fixes setup, hold, and logical design constraint violations and can optionally perform leakage-power, signal integrity, and area recovery optimization.

To perform standard postroute optimization, use the `route_opt` command, as described in this section.

- Focal optimization, which targets a single violation type.

To perform focal optimization, use the `focal_opt` command, as described in “[Performing Focal Optimization](#)” on page 6-104.

By default, the `route_opt` command routes the signal nets and then performs two passes of postroute optimization and ECO routing. If signal integrity mode is enabled, the first pass does not perform crosstalk reduction or signal integrity optimization, but the second pass does. If you have already routed the signal nets, use the `-skip_initial_route` option to perform postroute optimization without performing signal routing.

The `route_opt` command also provides an incremental postroute optimization capability, which performs only one pass of postroute optimization and ECO routing. If signal integrity mode is enabled, the tool performs crosstalk reduction and signal integrity optimization. To perform incremental postroute optimization, use the `-incremental` option with the `route_opt` command.

The following sections describe ways in which you can modify the postroute optimization behavior.

- [Performing Concurrent Clock and Data Optimization](#)
- [Controlling the Optimization Effort](#)
- [Fixing Hold Time Violations](#)
- [Performing Leakage-Power Optimization](#)
- [Performing Signal Integrity Optimization](#)
- [Performing Area Recovery](#)
- [Performing Register-to-Register Optimization](#)
- [Performing High-Resistance Optimization](#)
- [Controlling the Optimization Reporting](#)

Performing Concurrent Clock and Data Optimization

By default, the `route_opt` command optimizes only the data paths during postroute optimization. To enable the optimization of clock paths in addition to the data paths, use the `-concurrent_clock_and_data` or `-only_concurrent_clock_and_data` option. When you use the `-concurrent_clock_and_data` option, the tool optimizes the data paths to fix setup, hold, and logical design rule violations. When you use the `-only_concurrent_clock_and_data` option, the tool optimizes the data paths only to fix setup violations. When you use these options, you can control the scope of the optimization by using the `set_concurrent_clock_and_data_strategy` command. For more information, see [“Setting Up for Concurrent Clock and Data Optimization” on page 5-94](#).

Note:

You can use the concurrent clock and data optimization options only with the following `route_opt` options: `-effort`, `-xtalk_reduction`, `-power`, and `-incremental`.

Controlling the Optimization Effort

By default, the `route_opt` command uses medium effort during postroute optimization. You can change the effort level by using the `-effort` option. When you select high effort, the tool is more aggressive during optimization and runs three optimization loops.

In addition to controlling the overall effort level, you can control the effort used for specific optimizations.

- To enable additional aggressive fixing of hold time and maximum transition violations, set the `routeopt_enable_aggressive_optimization` variable to true.

```
icc_shell> set_app_var routeopt_enable_aggressive_optimization true
```

- To enable additional aggressive fixing of total negative slack, set the `-tns_effort` option of the `set_optimization_strategy` command to `high`.

```
icc_shell> set_optimization_strategy -tns_effort high
```

- To restrict total negative slack optimization to size-only optimization, set the `routeopt_restrict_tns_to_size_only` variable to true.

```
icc_shell> set_app_var routeopt_restrict_tns_to_size_only true
```

- To restrict all optimizations to size-only optimization, use the `-size_only` option with the `route_opt` command.

When you use the `-size_only` option, control the type of sizing performed by using the `-effort` option with the following settings:

- low, which footprint swapping
 - medium, which performs in-place size-only optimization
 - high, which performs cell sizing
- To restrict postroute optimization to footprint-preservation sizing to increase the drive strength of weak cells, set the `routeopt_only_size_weak_drive_cells` variable to true.

You can use this technique during incremental postroute optimization (`route_opt -incremental`) to generate a robust design with similar or better QoR that is less sensitive to variations.

Fixing Hold Time Violations

The `route_opt` command fixes hold time violations on those clocks that have a `fix_hold` attribute of `true`.

To disable hold fixing, use the following command to remove the `fix_hold` attribute from all clocks:

```
icc_shell> remove_attribute [get_clocks *] fix_hold
```

To enable hold fixing, use the `set_fix_hold` command to set the `fix_hold` attribute on the clocks on which to perform hold fixing.

For example, to enable hold fixing on clock `clk`, use the following command:

```
icc_shell> set_fix_hold clk
```

To enable hold fixing on all clocks, use the following command:

```
icc_shell> set_fix_hold [all_clocks]
```

Note:

If you enable hold fixing for a multicorner-multimode design, the tool considers the timing from driver to endpoint on a per-scenario basis and fixes hold violations accordingly.

You can specify a list of preferred buffers for hold fixing during postroute optimization by using the `set_prefer` and `set_fix_hold_options` commands. For example, the following commands instruct the tool to use cells `BUF1` and `BUF2` as the preferred cells during hold fixing. Note that cells `BUF1` and `BUF2` can also be used to resolve setup and DRC violations.

```
icc_shell> set_prefer -min {BUF1 BUF2}
icc_shell> set_fix_hold_options -preferred_buffer
```

If you use the `set_dont_use` command to set the `dont_use` attribute on cells `BUF1` and `BUF2` as shown in the following script, the tool uses cells `BUF1` and `BUF2` to fix hold violations, but not setup and DRC violations.

```
icc_shell> set_dont_use {BUF1 BUF2}
icc_shell> set_prefer -min {BUF1 BUF2}
icc_shell> set_fix_hold_options -preferred_buffer
```

The tool also provides the following variables that control hold fixing:

- `routeopt_allow_min_buffer_with_size_only`

By default, the tool cannot insert buffers during size-only optimization. When you set this variable to `true` and use the `-size_only` option when you run the `route_opt` command, the tool can insert buffers to fix hold violations.

- `routeopt_enable_aggressive_optimization`

This variable enables additional, aggressive fixing of hold violations.

- `psyn_onroute_disable_hold_fix`

This variable disables hold fixing, but preserves the minimum timing on clocks that have the `fix_hold` attribute.

When you perform incremental postroute optimization, you can restrict the optimization to hold fixing by using the `-only_hold_time` option with the `route_opt -incremental` command.

Performing Leakage-Power Optimization

To perform leakage-power optimization,

1. For a multicorner-multimode design, use the `set_scenario_options` command to select the leakage scenarios.

For more information about selecting the leakage scenarios, see “[Enabling and Disabling Scenarios for Specific Tasks](#)” on page 3-37.

2. (Optional) Enable power-aware postroute optimization by setting the `-power_aware_optimization` option of the `set_route_opt_strategy` command to `true`.

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`.

3. Enable leakage-power optimization and recovery by using the `-power` option when you run the `route_opt` command.

To perform power-aware postroute optimization without power recovery, run the `route_opt` command without the `-power` option.

To perform only power recovery during incremental postroute optimization, use the `-only_power_recovery` option with the `route_opt -incremental` command.

Performing Signal Integrity Optimization

To perform signal integrity optimization, set the signal integrity options as described in “[Setting Crosstalk Reduction Options](#)” on page 6-66 and use the `-xtalk_reduction` option when you run the `route_opt` command.

Note:

You cannot perform signal integrity optimization during incremental postroute optimization.

Performing Area Recovery

To perform area recovery, use the `-area_recovery` option when you run the `route_opt` command.

To perform only area recovery during incremental postroute optimization, use the `-only_area_recovery` option with the `route_opt -incremental` command.

Performing Register-to-Register Optimization

To restrict postroute optimization to register-to-register optimization, use the `-register_to_register` option with the `route_opt -incremental` option.

Note:

You can perform register-to-register optimization only during incremental postroute optimization or focal optimization.

By default, register-to-register optimization fixes setup, hold, and logical DRC violations. To restrict the optimization to a specific violation type, use the `-size_only`, `-only_hold_time`, or `-only_design_rule` option in addition to the `-incremental` and `-register_to_register` options.

Performing High-Resistance Optimization

In advanced-node designs, wire delays are dominated by increased net and via resistance. High-resistance optimization enables optimization techniques that reduce transition time and net delays by targeting load-dependent delays. For these types of designs, high-resistance optimization results in improved setup QoR and reduced area with minimal impact on hold QoR.

To enable high-resistance optimization, set the `-high_resistance` option of the `set_optimization_strategy` command to `true` before running the `route_opt` command.

When high-resistance optimization is enabled, timing-driven routing considers the wire resistance when inserting vias to reduce the wire resistance while maintaining the routability.

Controlling the Optimization Reporting

The `route_opt` command provides information about the optimization before, during, and after the optimization stage.

- Before optimization starts, the `route_opt` command generates a QoR report.
- During optimization, the `route_opt` command outputs information about the optimization process.

To get more information about the optimization process, you can enable verbose reporting by setting the `routeopt_verbose` variable. This variable uses bitwise operation to enable various reporting capabilities. For details about setting this variable, see [SolvNet article 032174, “Understanding the routeopt_verbose Variable Settings.”](#)

- After optimization completes, the `route_opt` command generates a QoR report.
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`.

Performing Focal Optimization

You can use the `focal_opt` command to perform aggressive, topology-based optimization on your postroute design that is focused on fixing either the setup, hold, or logical DRC violations that remain after the postroute optimization performed by the `route_opt` command. You can also use the `focal_opt` command to perform register-to-register optimization, crosstalk reduction, or final stage leakage recovery.

For each run, you must specify the focus of the optimization.

- To fix setup violations, use the `-setup_endpoints` option.

To fix all setup violations, set this option to `all`. To fix specific setup violations, specify the endpoints by using an endpoint file or by providing a collection that contains the endpoints. For information about the format of the endpoint file, see [“Endpoint File Format” on page 6-106.](#)

By default, the tool optimizes only the data paths. When you use the `-setup_endpoints all` option, you can enable concurrent optimization of the clock and data paths by using the `-concurrent_clock_and_data` option. When you use this option, you can control the scope of the optimization by setting the following options with the `set_concurrent_clock_and_data_strategy` command:

`-adjust_boundary_registers`, `-ignore_boundary_path_timing`,

`-ignore_path_groups`, and `-adjustment_limit`. For more information about these options, see “[Setting Up for Concurrent Clock and Data Optimization](#)” on page 5-94. For overall design convergence, perform a final pass of incremental routing and postroute optimization after concurrent clock and data optimization by using the `route_opt -incremental` command.

- To fix hold violations, use the `-hold_endpoints` option.

To fix all hold violations, set this option to `all`. To fix specific hold violations, specify the endpoints by using an endpoint file or by providing a collection that contains the endpoints. For information about the format of the endpoint file, see “[Endpoint File Format](#)” on page 6-106.

- To fix logical DRC violations on nets, use the `-drc_nets` option.

To fix DRC violations on all nets, set this option to `all`. To fix DRC violations on specific nets, specify the nets by using a net file or by providing a collection that contains the nets. For information about the format of the net file, see “[Net File Format](#)” on page 6-107.

- To fix logical DRC violations on pins, use the `-drc_pins` option.

To fix DRC violations on all pins, set this option to `all`. To fix DRC violations on specific pins, specify the pins by using an endpoint file or by providing a collection that contains the pins. For information about the format of the endpoint file, see “[Endpoint File Format](#)” on page 6-106.

- To perform register-to-register optimization, use the `-register_to_register` option.

By default, when you enable register-to-register optimization, the `focal_opt` command fixes setup, hold, and logical DRC violations. You can restrict the optimization to a specific violation type by using the `-setup_endpoints`, `-hold_endpoints`, `-drc_nets`, or `-drc_pins` option.

- To perform crosstalk reduction on specific nets, use the `-xtalk_reduction` option.

Specify the nets on which to perform crosstalk reduction by using a net file or by providing a collection that contains the nets. For information about the format of the net file, see “[Net File Format](#)” on page 6-107.

- To perform final stage leakage recovery, use the `-power` option.

When you perform final stage leakage recovery, you must enable exactly one leakage scenario. For information about leakage scenarios, see “[Enabling and Disabling Scenarios for Specific Tasks](#)” on page 3-37.

For more information about final stage leakage recovery, see “[Performing Final Stage Leakage-Power Recovery](#)” on page 6-107.

By default, the `focal_opt` command uses medium effort to fix the remaining violations. When using medium effort, the `focal_opt` command explores more solution spaces than the `route_opt` command while fixing the violations, including solutions that exceed the

density limit. If medium effort does not fix the remaining violations, you can use high effort (`-effort high` option), which explores even more solution spaces and uses runtime-expensive accurate signal integrity reestimation. Due to the additional runtime required for high effort, use this only when you have a few remaining violations to close. Note that the `-effort` option does not apply to final stage leakage recovery.

If your design has remaining violations, and you want to prioritize the fixing of those violations above all other cost priorities, you can use the `-prioritize` option. When you use the `-prioritize` option, the `focal_opt` command might violate other constraints to fix the prioritized violations. Note that the `-prioritize` option does not apply to crosstalk reduction or final stage leakage recovery.

If your design is very sensitive to postroute optimization changes, you can limit the optimizations to sizing-only optimizations by specifying the mode with the `-size_only_mode` option. When you specify this option you must select one of the following sizing modes: density-based sizing (`density`), in-place sizing (`in_place`), or footprint-preservation sizing (`footprint`). Note that the `-size_only_mode` option does not apply to crosstalk reduction or final stage leakage recovery.

If you find that the `focal_opt` command leaves unfixed violations, you can enable verbose reporting during optimization by setting the `routeopt_verbose` variable. This variable uses bitwise operation to enable various reporting capabilities. For details about setting this variable, see [SolvNet article 032174, “Understanding the routeopt_verbose Variable Settings.”](#)

Endpoint File Format

To fix setup, hold, or logical DRC violations on specific endpoints, you can specify the endpoints in an endpoint file. This file can be either uncompressed or compressed in gzip format. When the file is a gzip file, it must have the `.gz` file extension.

The following formats are supported for specifying an endpoint:

- Endpoint only

```
I_STACK_TOP/I3_STACK_MEM/Stack_Mem_reg_2__1_/D
```

- Endpoint with a slack value

```
I_STACK_TOP/I3_STACK_MEM/Stack_Mem_reg_2__1_/D 0.58      0.44 r      -0.14
```

By default, when you use this format with the `-setup_endpoints` or `-hold_endpoints` option, the tool uses the slack values specified in the endpoint file during focal optimization. To use the computed slack values instead, set the `focalopt_endpoint_margin` variable to `false`.

When you use this format with the `-drc_pins` option, the tool ignores the slack information and uses only the pin information.

The best way to generate these lines is by editing the file generated by the `report_constraint` command.

Net File Format

To fix logical DRC violations on specific nets, you must specify the nets in a net file.

The following formats are supported for specifying a net:

- Net name only
`I_STACK_TOP/n342`
- Net name with a violation

<code>I_STACK_TOP/n342 0.70</code>	0.89	-0.19	(VIOLATED)
------------------------------------	------	-------	------------

When you use this format with the `-drc_nets` option, the tool ignores the slack information and uses only the net information.

The best way to generate these lines is by editing the file generated by the `report_constraint` command.

Performing Final Stage Leakage-Power Recovery

Final stage leakage-power recovery optimizes the leakage power based on the `cell_footprint` attribute of the library cells. Ensuring that your design is close to timing closure is a prerequisite for final stage leakage-power recovery.

The final stage leakage-power recovery step considers library cells to have the same footprint if the cells have

- The same operating conditions (P, V, and T)
- Logical equivalence
- The same physical boundary, width, and height, as defined in the FRAM view
- The same `cell_footprint` attribute

For information about preparing the logic libraries for final stage leakage-power recovery, see “[Preparing Logic Libraries for Final Stage Leakage-Power Recovery](#)” on page 3-31.

To perform final stage leakage-power recovery on a postroute design,

1. For a multicorner-multimode design, use the `set_scenario_options` command to select the leakage scenario. You can select only a single leakage scenario.

To ensure that you set only one leakage scenario, use the following commands:

```
icc_shell> set leakage_scenario S1
icc_shell> set_scenario_options -leakage_power false \
    -scenarios [all_active_scenarios]
icc_shell> set_scenario_options -leakage_power true \
    -scenarios $leakage_scenario
```

If you do not have exactly one leakage scenario defined during final stage leakage-power recovery, the tool issues the PSYN-706 error message.

Error: There are multiple scenarios or no scenario with `set_scenario_options -leakage_power true`. (PSYN-706)

For more information about selecting the leakage scenarios, see “[Enabling and Disabling Scenarios for Specific Tasks](#)” on page 3-37.

2. If the postroute design has setup, hold, or logical DRC violations, run the `focal_opt` command to perform focal optimization to fix these violations.

For more information about the `focal_opt` command, see “[Performing Focal Optimization](#)” on page 6-104.

3. (Optional) Run the `verify_zrt_route` command to report the routing DRC violations that exist before final stage leakage-power recovery.

Perform this step to validate the cell footprint in the physical libraries by comparing the DRC violations reported before and after final stage leakage-power recovery.

4. Run the `focal_opt -power` command to perform postroute focal optimization for leakage-power recovery.

By default, this step uses footprint swapping to improve the leakage power on paths with positive slack. You can control the tradeoff between leakage-power recovery and timing QoR preservation by setting the `focalopt_power_critical_range` variable. This variable specifies the slack threshold for leakage-power recovery and timing QoR preservation. The `focal_opt -power` command performs leakage-power recovery only on cells whose worst slack is greater than the power critical range value and preserves the timing QoR such that the worst slack is greater than or equal to the power critical range value.

When you set this variable to a positive number, the command preserves timing QoR with positive slack and performs less leakage-power recovery. When you set this variable to a negative number, the command performs more aggressive leakage-power recovery and allows timing degradation. For multicorner-multimode designs, the tool uses only the leakage scenario for leakage-power recovery, but it considers all active scenarios for

timing QoR. Note that this command does not perform placement legalization or ECO routing after the leakage-power recovery.

When you use the `-power` option, you cannot use any other options with the `focal_opt` command.

When you run the `focal_opt -power` command, pay attention to the FOPT-28 and FOPT-29 messages in the log file.

Information: Optimizing leakage power with 3 voltage thresholds (FOPT-028).

Warning: Libcells cell_name1 and cell_name2 have same footprint but different pin geometry - eco route may be required after Leakage optimization (FOPT-029).

These information and warning messages list the library cells that have issues with the `cell_footprint` attributes.

5. (Optional) Run the `verify_zrt_route` command to report the routing DRC violations that exist after final stage leakage-power recovery.

Compare the reported violations to those reported before final stage leakage-power recovery. If the number of violations reported after running the `focal_opt -power` command is greater than the number of violations reported before, it implies that the physical libraries have issues such as dissimilar pin shape that result in additional DRC violations.

The following example script shows how to perform final stage leakage-power recovery for a multicorner-multimode design after routing optimization:

```
# Set one scenario as the leakage corner
set_scenario_options -leakage_power false \
    -scenarios [all_active_scenarios]
set_scenario_options -leakage_power true -scenarios S1

# Fix setup, hold, and logical DRC violations
focal_opt -setup_endpoints all
focal_opt -hold_endpoints all
focal_opt -drc_nets all

# Perform final stage leakage-power recovery
focal_opt -power
```

Performing ECO Routing

Whenever you modify the nets in your design, you need to run engineering change order (ECO) routing to reconnect the routing.

To run ECO routing, use the `route_zrt_eco` command (or choose Route > ECO Route in the GUI). The `route_zrt_eco` command sequentially performs global routing, track assignment, and detail routing to reconnect the routing. Zroute can reserve space for redundant vias during ECO routing. For information about this capability, see “[Concurrent Soft-Rule-Based Redundant Via Insertion](#)” on page 7-30.

By default, the `route_zrt_eco` command

- Considers timing and crosstalk during routing, which means that the tool performs extraction and updates the timing before performing the routing.

The extraction and timing update can be time consuming and might not be necessary for your design. To prevent the extraction and timing update, use the following commands to disable the timing-driven and crosstalk-driven modes:

```
icc_shell> set_route_zrt_global_options \
           -crosstalk_driven false -timing_driven false
icc_shell> set_route_zrt_track_options \
           -crosstalk_driven false -timing_driven false
icc_shell> set_route_zrt_detail_options -timing_driven false
```

- Works on all open nets in the design

To perform ECO routing only on specific nets, use the `-nets` option to specify the nets.

- Ignores existing global routes

To honor the existing global routes and perform incremental global routing, set the `-reuse_existing_global_route` option to `true`.

- Relieves congestion by reusing dangling wires instead of rerouting detail routed wires

To disable the reuse of dangling wires, set the `-utilize_dangling_wires` option to `false`.

- Performs a maximum of 40 detail routing iterations

To change the maximum number of detail routing iterations, use the `-max_detail_route_iterations` option.

- Fixes hard DRC violations on the entire design by rerouting any wires, whether modified or not
 - To fix DRC violations only in the neighborhood of the open nets, set the `-open_net_driven` option to `true`.

Note:

When you use the `-nets` option to perform ECO routing on specific nets, Zroute fixes DRC violations only within the bounding box of the specified nets. In this case, Zroute ignores the setting of the `-open_net_driven` option.

- To change the scope of rerouting performed by the ECO router, use the `-reroute` option.
 - To limit rerouting to modified wires, set this option to `modified_nets_only`.
 - To first attempt to fix DRC violations by rerouting modified nets and then reroute other nets if necessary, set this option to `modified_nets_first_then_others`.
A common use for this option is to route the clock nets affected by an ECO in a fully routed design.
- To enable the fixing of soft DRC violations, such as bridge rule violations, after the final detail routing iteration, use the following command:

```
icc_shell> set_route_zrt_common_options \
           -post_eco_route_fix_soft_violations true
```

Zroute generates a DRC violations summary at the end of each detail routing iteration. Before reporting the final DRC violations, Zroute merges redundant violations. For more information about the DRC violations reported by Zroute, see “[Performing Design Rule Checking Using the IC Compiler Tool](#)” on page 6-120.

Zroute also reports the nets changed during ECO routing. By default, it reports the first 100 changed nets. You can use the `-max_reported_nets` option to set a different limit on the reported nets. To report all changed nets, set the `-max_reported_nets` option to `-1`.

Cleaning Up Routed Nets

After routing is complete, you can clean up the routed nets by running the `remove_zrt_redundant_shapes` command (or by choosing Route > Remove Redundant Shapes in the GUI).

```
icc_shell> remove_zrt_redundant_shapes
```

By default, this command reads the DRC information stored in the CEL view of the design and then removes dangling and floating net shapes from all nets in the design based on this information. To run the `verify_zrt_route` command to get the DRC information instead of using the information stored in the CEL view, use the `-initial_drc_from_input false` option.

You can restrict the removal to

- Specific nets by using the `-nets` option
- Specific layers by using the `-layers` option
- Fixed or unfixed route types by using the `-route_types` option

When removing dangling net shapes, the tool does not change topologies or connections and does not touch terminals. In addition, no changes are made to open nets or nets with DRC violations.

You can disable the removal of dangling net shapes by using the `-remove_dangling_shapes false` option. You can disable the removal of floating net shapes by using the `-remove_floating_shapes false` option.

In addition to removing dangling and floating net shapes, this command can also remove loops in the specified nets. To remove loops, use the `-remove_loop_shapes true` option.

By default, the `remove_zrt_redundant_shapes` does not report the changes it makes. To report the changes, use the `-report_changed_nets true` option.

For example, to remove dangling net shapes, floating net shapes, and loops from the net named `my_net`, use the following command:

```
icc_shell> remove_zrt_redundant_shapes -nets my_net \
           -remove_loop_shapes true
```

After cleaning up the routed nets, reverify the routing, as described in “[Performing Design Rule Checking Using the IC Compiler Tool](#)” on page 6-120.

Saving the Routing Information

You can save the routing information by using the `write_route` command. This command generates a script file that contains the Tcl commands to generate the current routing. You must specify the name of the generated file by using the `-output` option.

```
icc_shell> write_route -output my_route.tcl
```

By default, the `write_route` command includes the following information in the generated Tcl file:

- The routes for all nets in the design

You can output the routes only for specific nets by using the `-nets` option or for specific wires and vias by using the `-objects` option.

- The route guides

If you do not want to save the route guide information, use the `-skip_route_guide` option.

By default, metal fill information is not written to the generated file. To include metal fill information, use the `-output_metal_fill` option.

Analyzing the Routing Results

You can analyze the routing results by reporting on the cell placement and routing statistics. The following sections describe how to perform these tasks:

- [Reporting Cell Placement and Routing Statistics](#)
- [Reporting Unfixed Timing Violations](#)
- [Analyzing Congestion](#)
- [Performing Design Rule Checking Using the IC Compiler Tool](#)
- [Performing Signoff Design Rule Checking](#)
- [Analyzing DRC Violations](#)

Reporting Cell Placement and Routing Statistics

You can report on cell placement and routing statistics to help you determine whether to perform further optimizations to improve timing. For example, if the statistics show that an area has high congestion, an additional optimization might not have room to add or size up standard cells.

To view the place and route summary report, run the `report_design_physical -verbose` command.

Reporting Unfixed Timing Violations

To report the causes for unfixed hold time or maximum transition time violations in the postroute design, run the `analyze_design_violations -stage postroute` command. For detailed information about running this command, see “[Reporting Unfixed Violations](#)” on [page 4-53](#).

Analyzing Congestion

To help you analyze the congestion in your design, the tool can generate both an ASCII congestion report, as shown in [Example 6-5](#) and a visual congestion map, as shown in [Figure 6-8 on page 6-118](#). You can generate the congestion report and congestion map for each routing stage: global routing, track assignment, and detail routing. The more routing stages that have been completed, the more accurate the congestion information is; however, the harder it is to fix the congestion issues.

Generating a Congestion Report

To generate a congestion report, run the `report_congestion` command.

```
icc_shell> report_congestion
```

By default, this command generates a global route congestion report by running global routing with minimum effort and reports a summary of the overflow information for the entire design, as shown in [Example 6-5](#).

Example 6-5 Default Global Route Congestion Report

```
Information: Reporting global route congestion data from Milkyway...
```

```
Both Dirs: Overflow = 39 Max = 8 (1 GRCs) GRCs = 14 (0.26%)
H routing: Overflow = 6 Max = 2 (2 GRCs) GRCs = 4 (0.07%)
V routing: Overflow = 33 Max = 8 (1 GRCs) GRCs = 10 (0.18%)
```

In the default congestion report,

- “H routing” refers to results for horizontal routes only and “V routing” refers to results for vertical routes only.
- The Overflow value is the total number of wires in the design that do not have a corresponding track available.

By default, all global routing cells (GRCs) are included in the overflow calculation. To exclude the global routing cells that are blocked for routing, use the `set_route_zrt_global_options` command to set the `-exclude_blocked_gcells_from_congestion_report` global route option to `true`. In instances where the global router has to route through blocked global routing cells to find a solution, those global route cells are counted in the overflow number but not in the total number.

- The Max value corresponds to the highest number of overutilized wires in a single global routing cell.
- The GRCs value is the total number of overcongested global routing cells in the design.

The generated congestion information is stored in the Milkyway design database; however, the global routing results are not. By default, the congestion data is stored for each layer, which enables the tool to generate a layer-based congestion map; however, saving the layer-based information increases the size of the Milkyway design database. If you do not need layer-based congestion maps, you can disable this capability and save only aggregate congestion information by using the `set_route_zrt_global_options` command to set the `-layer_based_congestion_map` global route option to `false`.

To display the congestion report in the design database instead of regenerating a congestion report, use the `-from_saved_congestion_map` option when you run the `report_congestion` command.

Important:

Before using the `-from_saved_congestion_map` option, you must ensure that the existing congestion map is valid and up to date. If the design has changed since you generated the congestion map, you must regenerate the congestion report and cannot use this option.

You can report additional information in the congestion report by using the options shown in [Table 6-9](#).

Table 6-9 Generating Additional Congestion Information

Option	Description
-grc_based	Provides detailed information about the 10 worst global routing cells for each category: total, horizontal, and vertical.
-grc_number	Provides detailed information about the specified number of worst global routing cells for each category.
-overflow_threshold	Provides detailed information about the global routing cells that exceed the specified overflow threshold, up to a maximum of 10 global routing cells for each category.
-by_layer	Generates a congestion report for each routing layer. The type of report, summary or GRC-based, generated for each layer depends on the other options that you specify.

[Example 6-6](#) shows a GRC-based congestion report. This same format is used when you specify the -grc_based, -grc_number, or -overflow_threshold options; the only difference is the number of global routing cells reported in each category.

Example 6-6 GRC-Based Global Route Congestion Report

Information: Reporting global route congestion data from Milkyway...

```
There are 2704 GRCs (Global Route Cell) in this design
*****
***** GRC based congestion report *****
*****
14 GRCs with overflow: 4 with H overflow and 10 with V overflow

++++++ Top 10 congested GRCs +++++

GRC {624345 676005 650175 701835}:
  H routing capacity/demand = 81/81 (occupy rate = 1.00),
  V routing capacity/demand = 54/62 (occupy rate = 1.15) with overflow 8
GRC {624345 701835 650175 727665}:
  H routing capacity/demand = 74/74 (occupy rate = 1.00),
  V routing capacity/demand = 54/59 (occupy rate = 1.09) with overflow 5
...
GRC {676005 391875 701835 417705}:
  H routing capacity/demand = 12/14 (occupy rate = 1.17),
  V routing capacity/demand = 3/3 (occupy rate = 1.00) with overflow 2
```

```
++++++ Top 10 horizontally congested GRCs ++++++
GRC {650175 391875 676005 417705}:
  H routing capacity/demand = 12/14 (occupy rate = 1.17) with overflow 2
GRC {676005 391875 701835 417705}:
  H routing capacity/demand = 12/14 (occupy rate = 1.17) with overflow 2
...
GRC {727665 882645 753495 908475}:
  H routing capacity/demand = 91/91 (occupy rate = 1.00)
++++++ Top 10 vertically congested GRCs ++++++
GRC {624345 676005 650175 701835}:
  V routing capacity/demand = 54/62 (occupy rate = 1.15) with overflow 8
GRC {624345 701835 650175 727665}:
  V routing capacity/demand = 54/59 (occupy rate = 1.09) with overflow 5
```

In the GRC-based congestion report,

- The numbers after GRC represent the location of that global routing cell. For example, GRC {727665 882645 753495 908475} refers to the global routing cell with its lower-left corner at (727665, 882645) and its upper-right corner at (753495, 908475). To get the layout coordinates, divide the numbers by the precision value specified in the Milkyway design library.
- “H routing” refers to results for horizontal routes only, while “V routing” refers to results for vertical routes only.
- The “occupy rate” for a global routing cell is its demand divided by its capacity, where the capacity is the number of available routing tracks in a global routing cell and the demand is the number of nets that cross edge of that global routing cell.
- The overflow value is the total number of wires in the design that do not have a corresponding track available (demand - capacity).

You can generate a congestion report for a portion of the design by using the `-coordinates` option to specify a rectangular area or the `-polygon` option to specify a rectilinear area.

By default, the `report_congestion` command uses minimum effort when running global routing. You can increase the global routing effort by using the `-effort` option to specify a higher effort level of `low`, `medium`, or `high`.

You can also generate congestion reports based on the track assignment and detail routing in the design by setting the `-routing_stage` option to `track` or `detail`, respectively. When you generate these reports, the tool does not run track assignment or detail routing; it uses the existing information in the design. If the track assignment or detail route information does not already exist in the design database, the tool issues an error message and does not generate a congestion report.

Generating a Congestion Map

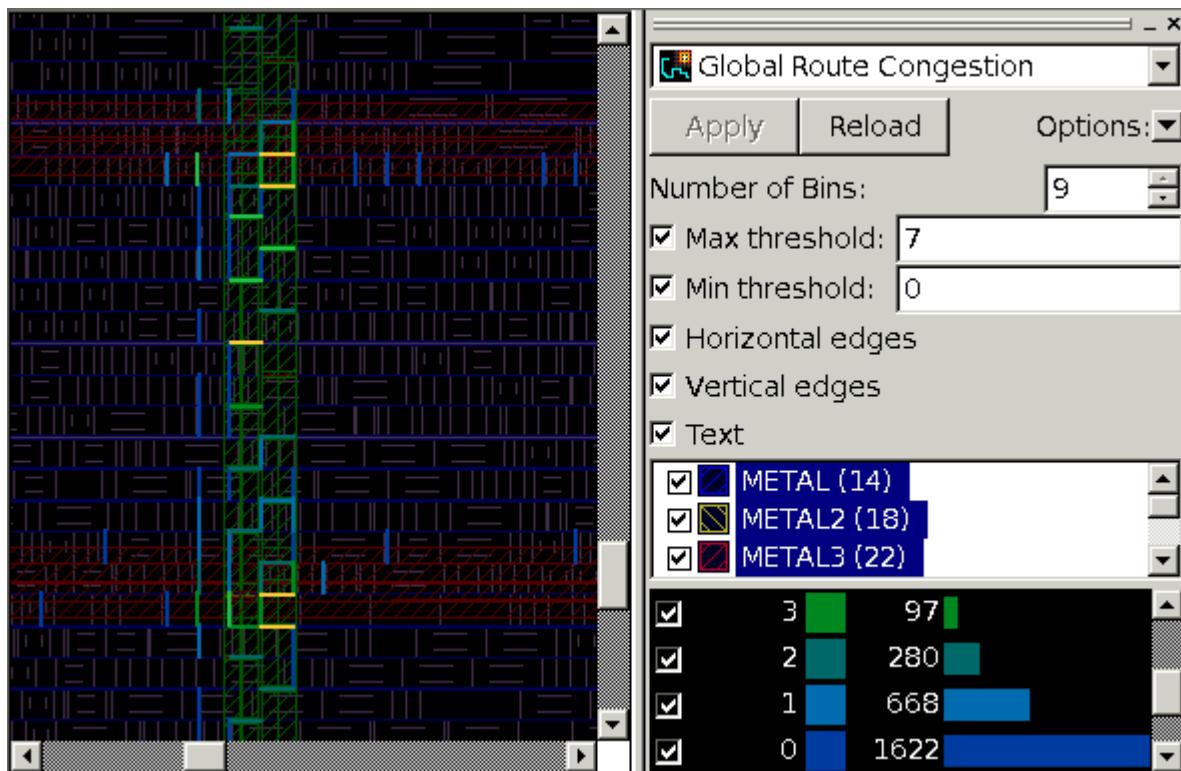
To display the global route congestion map, choose Route > Global Route Congestion Map in the GUI. If the design database contains global route congestion information, the tool generates the congestion map based on this information; otherwise, you must click Reload to generate the congestion map. When you click Reload, the tool opens a dialog box that contains the following command:

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

When you click OK in this dialog box, the tool generates a new congestion map. If you want to use different options for the `report_congestion` command, you can modify this command before clicking OK.

[Figure 6-8](#) shows an example of a congestion map.

Figure 6-8 Global Route Congestion Map



The congestion map shows the borders between global routing cells highlighted with different colors that represent different levels of overflow. If you select multiple layers, the overflow values shown in the congestion map represent the total overflow for all selected layers. Underflow is not considered; if a layer has underflow, it contributes zero overflow to

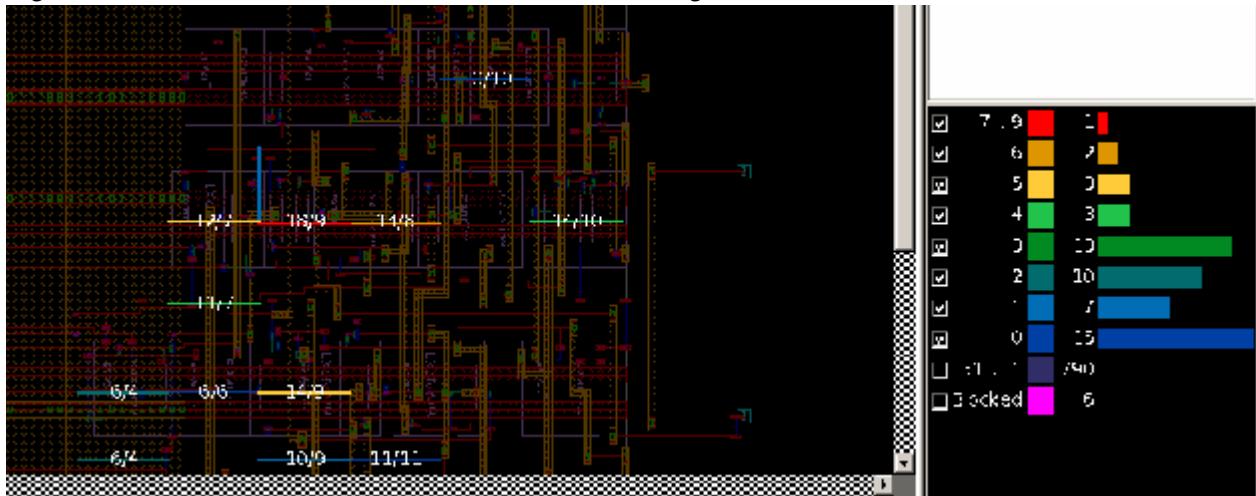
the total overflow calculation. If you select a single layer, the exact overflow or underflow information is displayed.

By default, all metal layers are selected in the congestion map, except those specified as ignored layers with the `set_ignored_layers` command. To display the congestion map for a subset of layers, select (or deselect) the layers on the Map Mode panel. For example, if the global routing report shows that the maximum overflow occurs on metal layer 2, you can deselect all layers, except for metal 2, to display only the metal 2 congestion.

The Map Mode panel also displays a histogram showing the number of global routing cells in different ranges (bins) of overflow values for the selected layers. If your design contains global routing cells that have no available routing resources, an additional bin named Blocked is displayed that shows the number of global routing cells with no routing resources. You can select which bins to display in the congestion map by selecting or deselecting them on the Map Mode panel.

If the design shows congested areas, zoom into the congested area to see the overflow number on the global routing cell. For example, in [Figure 6-9](#), the red highlight on the edge of the global routing cell shows 18/9. This means there are 9 wire tracks available, but 18 tracks are needed.

Figure 6-9 Global Route Overflow on Global Routing Cell



You can also generate congestion maps based on the track assignment and detail routing in your design. To generate a congestion map based on track assignment, choose **Route > Track Assign Congestion Map** in the GUI. To generate a congestion map based on detail routing, choose **Route > Detail Route Congestion Map** in the GUI. When you generate these congestion maps, the tool does not run track assignment or detail route information; it uses the existing information in the design. If the track assignment or detail route information does not already exist in the design database, the tool issues an error message and does not generate a congestion map.

Performing Design Rule Checking Using the IC Compiler Tool

To use the IC Compiler DRC engine to check the routing design rules defined in the Milkyway technology file, run the `verify_zrt_route` command (or choose Route > Verify Route in the GUI).

Note:

Do not run the `verify_route` command on a design routed with Zroute. The `verify_route` command is based on the classic router, which uses center lines to determine connectivity. Because Zroute does not use center lines to determine connectivity, you will see many DRC violations due to the difference in connectivity models.

By default, the `verify_zrt_route` command checks for routing DRC violations, unconnected nets, antenna rule violations, and voltage area violations on all routed signal nets in the design, except those marked as user nets, frozen nets, and PG nets.

To disable checks for routing DRC violations, set the `-drc` option to `false`. To disable checks for unconnected nets, set the `-open_net` option to `false`. To disable checks for antenna rule violations, set the `-antenna` option to `false`. To disable checks for voltage area violations, set the `-voltage_area` option to `false`.

To verify the routing only for specific nets, specify the nets by using the `-nets` option. To check user routes, set the `-check_from_user_shapes` option to `true`. To check frozen routes, set the `-check_from_frozen_shapes` option to `true`.

To save time, you can restrict the routing verification to specific regions of the design by using the `-coordinates` option to specify the lower-left and upper-right coordinates for each rectangular region. When you perform area-based DRC, the `verify_zrt_route` command checks only for DRC violations and voltage area violations. It does not check for unconnected nets, antenna violations, or tie-to-rail violations, as these are net-based violations.

Note:

The `-coordinates` option and the `-nets` option are mutually exclusive; you can use only one of these options.

The `verify_zrt_route` command reports the following DRC violations:

- Spacing violations
 - Different-net wire spacing
 - Different-net nondefault wire spacing (note that the DRC report refers to nondefault routing rules as variable rules)
 - Different-net via-cut spacing

- Different-net nondefault via-cut spacing (note that the DRC report refers to nondefault routing rules as variable rules)
- Different-net fat extension spacing
- Dog bone spacing
- End-of-line spacing
- Enclosed via spacing
- Same-net spacing
- Same-net via-cut spacing
- Same-net fat extension spacing
- Special notch spacing
- U-shape spacing
- Via-cut to metal spacing
- Soft spacing
- Area violations
 - Less than minimum area
 - Less than minimum enclosed area
 - Fat wire via keepout area
 - Jog wire via keepout area
- Length and width violations
 - Less than minimum width
 - Less than minimum length
 - Less than minimum edge length
 - Protrusion length
- Contact violations
 - Needs fat contact
 - Needs poly contact
 - Needs fat contact on extension
 - Over maximum stack level

- Enclosure violations
 - End-of-line wire via enclosure
 - Jog wire via enclosure
 - T-shape wire via enclosure
- Others
 - Open nets, except when doing area-based DRC

By default, the `verify_zrt_route` command reports a maximum of 200 open nets. To report all open nets, use the `-report_all_open_nets true` option (or select “Report all open nets” in the GUI).
 - Antenna violations, except when doing area-based DRC
 - Nets crossing the top-cell boundary
 - Frozen layers
 - Minimum layer
 - Maximum layer
 - Voltage area violations

Note:

These violations are also reported after each detail route iteration.

After you run the `verify_zrt_route` command, you can use the DRC query commands to get more information about the violations or use the error browser to examine the violations in the GUI. For information about analyzing the DRC violations, see [“Analyzing DRC Violations” on page 6-133](#).

After running `verify_zrt_route`, you can use the following command to run incremental detail routing that uses the `verify_zrt_route` results as input:

```
icc_shell> route_zrt_detail -incremental true \
           -initial_drc_from_input true
```

Note:

Incremental detail routing does not fix open nets. To fix open nets, you must run ECO routing. For information about ECO routing, see the next section, [“Performing ECO Routing”](#).

Performing Signoff Design Rule Checking

Signoff design rule checking runs the IC Validator tool within the IC Compiler tool to check the routing design rules defined in the foundry runset. To perform signoff design rule checking, run the `signoff_drc` command (or choose Verification > Signoff DRC in the GUI). In addition, you can use the `signoff_autofix_drc` command to automatically fix the DRC violations detected by the `signoff_drc` command. For more information, see “[Automatically Fixing Signoff DRC Violations](#)” on page 6-130.

Note:

An IC Validator license is required to run the `signoff_drc` command.

To use `signoff_drc` to perform signoff design rule checking,

- Set up the validation tool environment.
- Set up the physical signoff options.
- Run the `signoff_drc` command (or choose Verification > Signoff DRC in the GUI).

The following sections describe these tasks.

Setting Up the IC Validator Environment

To run the `signoff_drc` command, you must specify the location of the IC Validator executable by setting the `ICV_HOME_DIR` environment variable. You can set this variable in your `.cshrc` file. To specify the location of the IC Validator executable, use commands similar to those shown in the following example:

```
setenv ICV_HOME_DIR /root_dir/icv  
set path = ($path $ICV_HOME_DIR/bin/AMD.64)
```

You must ensure that the version of the IC Validator executable that you specify is compatible with the IC Compiler version that you are using.

For more information about the IC Validator tool, see the IC Validator documentation, which is available on SolvNet.

Setting The Physical Signoff Options

To set the physical signoff options, use the `set_physical_signoff_options` command (or choose Verification > Set Physical Signoff Options in the GUI).

[Table 6-10](#) shows the physical signoff options that apply to the `signoff_drc` command.

Table 6-10 Physical Signoff Options for signoff_drc

Command option	Description
<code>-exec_cmd icv</code> ("Executable name" box in the GUI)	Specifies the validation tool to use for design rule checking (optional). If you do not specify this option, the <code>signoff_drc</code> command uses the value stored in the Milkyway design library. If there is no value stored in the Milkyway design library, an error occurs.
<code>-drc_runset file_name</code> (DRC box in the GUI "Foundry runset" section)	Specifies the foundry runset to use for design rule checking (required).
<code>-mapfile file_name</code> ("Layer mapping file" box in the GUI)	Specifies the layer mapping file (optional). For more information, see " Defining the Layer Mapping Between Milkyway and the Foundry Runset ."

To report the current settings for the physical signoff options, use the `report_physical_signoff_options` command.

Defining the Layer Mapping Between Milkyway and the Foundry Runset

In general, the Milkyway technology file and the foundry runset file used by the IC Validator tool use the same layer numbers. If they do not, you must supply a layer mapping file to map the Milkyway layers to the layers used in the runset file.

The `signoff_drc` command accepts a layer-mapping file in IC Validator or Milkyway format.

- The syntax of the mapping information in an IC Validator mapping file is

Milkyway_layer_list > runset_layer_list

The rules for specifying the layer lists are

- Each layer in a layer list must be separated by a space. Or, if specified in a range, the range must consist of a starting layer and an ending layer joined by a dash (-).
- Generally, parentheses should surround the layer list. However, you can omit parentheses if you specify only one layer.

You can include comments in a layer mapping file by preceding the comment with a pound sign (#). All text that follows a pound sign on a given line is a comment.

- The syntax of the mapping information in a Milkyway mapping file is

MilkywayObjType MilkywayLayer[:DataType] RunsetLayer[:RunsetDataType]

The *MilkywayObjType* argument is a string of one to three characters that identifies the Milkyway objects to be translated. The required first character is the code for the type of Milkyway object to be translated: A for all, T for text, or D for data. The optional second character specifies the net type, such as S for signal, P for power, or G for ground. An optional third character specifies the pin code. The Milkyway layer is a name or an integer. The runset layers and data types are integers.

You can add a comment to the file by inserting a semicolon. Text is ignored from the semicolon to the end of the line.

You can save the Milkyway layer mapping file in the Milkyway design library by using the `set_stream_layer_map_file -format out` command. When the Milkyway design library contains a layer mapping file, it defines the default mapping. The layer mapping file specified by the `set_physical_signoff_options -mapfile` option overrides the layer mapping file saved in the Milkyway design library.

For more information about the Milkyway mapping file, see the “Layer Mapping File: Milkyway to GDSII or OASIS” section in Chapter 3 of the *Library Data Preparation for IC Compiler User Guide*.

Performing Distributed Design Rule Checking

By default, the `signoff_drc` command uses a single process to perform design rule checking. To reduce the turnaround time used for design rule checking, you can use distributed processing. To enable distributed processing, you must define the distributed processing configuration by using the `set_host_options` command.

If you have defined more than one distributed processing configuration with the `set_host_options` command, the `signoff_drc` command selects the IC Validator processing method in the following order of priority:

1. Job submission through a user-defined distributed processing script

To enable job submission using your own script with the `set_host_options` command, use the `-submit_command` option to specify the location of your job submission script.

For example, to specify a configuration named `custom4` that enables a maximum of four processes using your job submission script, use the following command:

```
icc_shell> set_host_options -name custom4 -num_processes 4 \
    -submit_command /usr/local/bin/my_submit_command
```

2. Job submission through the Load Sharing Facility (LSF) or the Sun Grid Engine (SGE)

To enable LSF or SGE job submission with the `set_host_options` command, use the `-pool` option to specify the mode and the `-num_processes` option to specify the maximum number of processes.

For example, to specify a configuration named `lsf4` that enables a maximum of four processes using LSF, use the following command:

```
icc_shell> set_host_options -name lsf4 -pool lsf -num_processes 4
```

To specify a configuration named `grd4` that enables a maximum of four processes using SGE, use the following command:

```
icc_shell> set_host_options -name grd4 -pool grd -num_processes 4
```

3. Distributed processing on the specified hosts

To enable distributed processing with the `set_host_options` command, specify the hosts and use the `-num_processes` option to specify the maximum number of processes on each host. For example, to specify a configuration named `dp4` that enables a maximum of four processes, with a maximum of two processes each on `machineA` and `machineB`, use the following command:

```
icc_shell> set_host_options -name dp4 -num_processes 2 \
    {machineA machineB}
```

Note:

If you define only a multithreading configuration for (`set_host_options -max_cores`), the `signoff_drc` command uses distributed processing on the current machine for the IC Validator run.

To ensure that you are using the intended distributed processing configuration, remove the current configurations by using the `remove_host_options` command before defining the distributed processing configuration for the `signoff_drc` command. To report the current distributed processing configurations, use the `report_host_options` command.

For more information about the `set_host_options` command, see “[Enabling Multicore Processing](#)” on page 2-12.

Running the `signoff_drc` Command

By default, the `signoff_drc` command uses the FRAM view to check all cell types (standard cells, macro cells, and I/O pad cells) on all routing layers of the entire chip for all rules specified in the foundry runset. To use the CEL view instead of the FRAM view, use the `-read_cel_view` option or select the “CEL view” option in the GUI. Using the CEL view can expose problems that are masked by the FRAM view abstraction.

Note:

The `signoff_drc` command uses the on-disk design information, not the design information in memory. You must save the current state of the design before running the `signoff_drc` command.

You can specify additional options for the IC Validator command line by using the `-user_defined_options` option when you run the `signoff_drc` command. The string that you specify in this option is added to the command line used to invoke design rule checking in the IC Validator tool. The IC Compiler tool does not perform any checking on the specified string.

By default, the IC Validator tool checks for both top-level and child-level errors and reports a maximum of 1000 DRC errors per rule. To ignore child-level errors, use the `-ignore_child_cell_errors` option. You can override the maximum error count by using the `-max_errors_per_rule` option when you run the `signoff_drc` command.

You can restrict the design rule checking to specific areas of the chip, to specific design rules, to specific layers, and to specific cell types.

- To restrict the checking to specific areas of the chip,
 - If you are using the command line, use the `-bounding_boxes` option to specify the coordinates of each area to check. To exclude checking in specific areas, use the `-excluded_bounding_boxes` option. For either option, you can specify multiple areas by specifying the coordinates for each area.
 - If you are using the GUI, specify the coordinates for each area to check in the “Included bounding boxes” list and specify the coordinates for each area to exclude from checking in the “Excluded bounding boxes” list. To specify an area, either draw the bounding box or enter the x- and y-coordinates of the box. For each area, you can also enable or disable snapping. If snapping is enabled, you can specify that the

bounding box should snap to the minimum grid (the default), placement site, routing track, middle routing track, or user grid.

- If you have previously run the `signoff_drc` command on a design, use the `-auto_eco` option to restrict the checking to those areas of the design that have been modified since the last run.

By default, the tool compares the current design to the version used for the previous signoff DRC run. To compare the current design to a different design, use the `-pre_eco_mw_cel` option to specify the comparison design. The tool gets the information required for incremental change detection from the error view named `cell_sdrc.err`, where `cell` is either the current design name or the name specified by the `-pre_eco_mw_cel` option.

- To restrict the checking to specific rules,
 - To check only the specified rules, specify the rules in the `-select_rule` option (or in the Pattern box in the “To use” section of the Rules area in the GUI).
 - To exclude checks for specific rule, specify the rules in the `-unselect_rule` option (or in the Pattern box in the Excluded section of the Rules area in the GUI).

In either case, you specify the rules by specifying a matching pattern for the rule names. The rule names are specified in the COMMENT section in the runset file. You can use these options together to customize the set of rules checked by the `signoff_drc` command.

For example, to restrict the `signoff_drc` command to route validation, select the metal layer rules and exclude the metal density rules.

- To restrict the checking to specific layers,
 - Specify the layers in the `-select_layers` option (or in the list associated with the “Selected layers” option in the “Check DRC violations on” area in the GUI).
- To restrict the checking to specific cell types,
 - Specify the cell types to exclude in the `-excluded_cell_types` option (or select the associated check box in the “Excluded cell types” area in the GUI).

After you complete design rule checking on the routing layers, you can perform a quick design rule check on the Milkyway database by rerunning the `signoff_drc` command with the `-check_all_layers` option. If you are using the GUI, set this option by selecting the “All runset layers (routing and device layers)” option in the “Check DRC violations on” area.

When you use this option, the design information comes from the CEL view and the validation tool checks all layers, including the nonrouting layers.

The `signoff_drc` command creates a Milkyway error view that you can use to report or display the DRC violations. Child-level errors are shown on only one of the cell instances, which makes it easier to identify lower-level errors. By default, the error view generated by

the `signoff_drc` command is saved in a file called `design_sdrc.err`. You can control this name by using the `-error_view` option (or the “Error cell name” box in the GUI).

To make it easier to view specific violations in the error browser, you can enable error categorization for the generated error view. When you enable this feature, the violations in the error view are annotated with their associated net type and route type. You can use these annotations to filter the violations displayed in the error browser. To enable this feature, use the `-categorize_errors true` option when you run the `signoff_drc` command. Note that using this feature slightly increases the runtime for the `signoff_drc` command. For information about using the error view to debug DRC violations, see [“Analyzing DRC Violations” on page 6-133](#).

In addition to the error view file, the `signoff_drc` command generates the following files, which you can use for debugging, and places them in a directory called `signoff_drc_run` under the current working directory:

- `design.errsum` or `design.LAYOUT_ERRORS`
These files contain an error summary report.
- `sdrc.ev`
This file contains the runset file, including any options added by the `signoff_drc` command.
- `sdrc.rc`
This file contains the IC Validator environment setup specified by the `signoff_drc` command options.
- `sdrc.log`
This file contains any errors that occurred during the `signoff_drc` run.
- `./run_details/design.sum`
This file contains the detailed log file from the validation tool.
- `./run_details/runset.dp.log`
This file contains information about the runset used to run signoff DRC. It is used as input to the script that generates the configuration file for automatic signoff DRC fixing.

You can control the directory location by using the `-run_dir` option (or the “Run directory” box in the GUI). You can specify either a relative path, in which case the directory is created under the current working directory, or an absolute path.

Automatically Fixing Signoff DRC Violations

Zroute can use the signoff DRC results to automatically fix the detected design rule violations. To use this capability, you must use version E-2010.12 or later of the IC Validator tool. For best results, use version F-2011.09-SP1 or later of the IC Validator tool.

To perform automatic fixing of the signoff DRC violations,

1. Set up the IC Validator environment as described in [“Setting Up the IC Validator Environment” on page 6-123](#).
2. Set up the physical signoff options as described in [“Setting The Physical Signoff Options” on page 6-124](#).
3. Run the `signoff_drc` command as described in [“Running the signoff_drc Command” on page 6-127](#).
4. (Optional) Generate a configuration file that aids in the fixing process.

The configuration file defines the layer-to-DRC-rule mapping. The `signoff_autofix_drc` command processes only those rules that are specified in the configuration file.

By default, the `signoff_autofix_drc` command automatically generates the configuration file. You can also manually generate a configuration file, as described in [“Creating an Autofix Configuration File” on page 6-132](#).

5. Run the `signoff_autofix_drc` command (or choose Verification > Signoff Autofix DRC in the GUI).

When you run the `signoff_autofix_drc` command, you must specify the location of the signoff DRC results from the previous run. To specify the directory that contains the IC Validator signoff DRC results, use the `-init_drc_error_db` option.

By default, the `signoff_autofix_drc` command runs two repair loops. In each loop, the IC Compiler tool

- a. Fixes the DRC violations based on the previous IC Validator signoff DRC results

For the first loop, regardless of its loop number, this is the IC Validator data in the directory specified by the `-init_drc_error_db` option. For successive loops, this is the IC Validator DRC data from the previous loop, which is stored in the directory specified by the `-run_dir` option (or the `signoff_autofix_drc_run` directory if you do not use this option).

By default, the `signoff_autofix_drc` command

- Tries to fix all design rule violations detected in the previous signoff DRC run

By default, if there are more than 1000 violations of a particular design rule, the `signoff_autofix_drc` command ignores that design rule and does not try to fix

those violations. To change the error threshold for ignoring a design rule, use the `-max_errors_per_rule` option.

To reduce runtime, you can ignore the DRC violations detected by the `verify_zrt_route` command by setting the `-ignore_route_drc` option to `true`. These violations can typically be fixed by Zroute detail routing instead of using IC Validator. By default, when you ignore the DRC violations detected by Zroute, the `signoff_ autofix_drc` command runs the `verify_zrt_route` command to identify these violations. To use the violations stored in the CEL view instead of running the `verify_zrt_route` command, set the `-initial_route_drc_from_input` option to `true`.

- Performs DRC fixing on the whole design

To reduce runtime, you can perform DRC fixing only in the local area of the DRC violations reported by the IC Validator tool by setting the `-incremental_level` option to `low`.

By default, the tool targets data nets for DRC fixing. To target clock nets instead, set the `-target_clock_nets` option to `true`.

- Performs five iterations of detail routing after DRC fixing to fix any DRC violations introduced during DRC fixing

If five iterations is not sufficient to fix DRC violations introduced during DRC fixing, use the `-max_detail_route_iterations` option to increase the number of iterations.

- b. Stores the modified design in a CEL view named `design_ADR_#`
- c. Runs IC Validator signoff DRC on the modified design

By default, when the `signoff_ autofix_drc` command runs signoff DRC checking, it reads the CEL views of the child cells in the design, performs signoff DRC checking only on those portions of the design affected by design rule fixing, and checks all rules specified in the foundry runset.

When signoff DRC checking reads the CEL view, it checks for violations inside the child cells in addition to the routes between the child cells. To check for violations only in the routes between the child cells, use the `-read_fram_view_only` option to force the signoff DRC checking to read the FRAM view instead of the CEL view.

For greater accuracy (at a cost of higher runtime), you can run signoff DRC on the entire design for one or both of the repair loops.

- To run signoff DRC on the entire chip only for the last repair loop, set the `-last_repair_loop_run_drc_on_full_chip` option to `true`.
- To run signoff DRC on the entire chip for both repair loops, set the `-incremental_level` option to `off`. Note that this setting can greatly increase runtime.

You can restrict the checking to specific rules by using the `-select_rule` option to specify rules to check and the `-unselect_rule` option to specify rules to exclude from checking. In either case, you specify the rules by specifying a matching pattern for the rule names. The rule names are specified in the COMMENT section in the runset file. You can use these options together to customize the set of rules checked by the `signoff_autofix_drc` command.

You can specify additional options for the IC Validator command line by using the `-user_defined_options` option when you run the `signoff_autofix_drc` command. The string that you specify in this option is added to the command line used to invoke design rule checking in the IC Validator tool. The IC Compiler tool does not perform any checking on the specified string.

- d. Stores the error information in an error view named `design_ADR_#_sdrc.err`

Note:

The error view generated for the last loop is called `design_ADR_sdrc.err`.

- e. Stores the IC Validator results in the directory specified by the `-run_dir` option (or the `signoff_autofix_drc_run` directory if you do not use this option)

To reduce the overall runtime for this flow, you can streamline the data transfer between the IC Compiler tool and the IC Validator tool by setting the `-incremental_level` option to `high`. This setting passes only the relevant design information between tools, rather than passing all design information. In addition, it performs DRC fixing only in the local area of the DRC violations reported by the IC Validator tool.

You can change the starting loop by using the `-start_repair_loop` option and change the ending loop by using the `-end_repair_loop` option. The number of the ending loop must always be greater than or equal to the number of the starting repair loop. By default, only the results of the ending loop are retained. To retain the results of all repair loops for debug purposes, use the `-keep_repair_loop_data true` option. Note that you cannot use this option when the `-incremental_level` option is set to `high`.

Creating an Autofix Configuration File

The autofix configuration file has the following format:

```
"mask_layer_name" "full_IC_Validator_DRC_rule_comment"
```

You can put comments in the configuration file by starting the line with the `#` character.

To generate the configuration file, run the `$ICV_HOME_DIR/contrib/generate_layer_rule_map.pl` script. This script has the following syntax:

```
generate_layer_rule_map.pl
-dplog IC_Validator_log_file
-tech_file Milkyway_technology_file
-o config_file_name
```

The IC Validator log file that you specify in the `-dplog` option is the log file that was generated by the initial `signoff_drc` run. This file is called `runset.dp.log` and is located in the `run_details` subdirectory of the `signoff_drc_run` directory (or the directory specified by the `-run_dir` option of the `signoff_drc` command).

To specify the location of the configuration file you created, use the `-config_file` option with the `signoff_autofix_drc` command.

Verifying the Routing in an External Tool

You can perform design rule checking with the Calibre tool, convert the Calibre DRC error file to a Milkyway error view, and then report or view the errors in the IC Compiler tool. To convert the Calibre DRC error file to a Milkyway error view, use the `read_drc_error_file` command (or choose Verification > Read Third-party DRC Error File in the GUI).

For example,

```
icc_shell> read_drc_error_file Calibre_error_file
```

By default, the command generates a Milkyway error view called `cell_name.err`, where `cell_name` is derived from the Calibre error report. You can specify a name for the Milkyway error view by using the `-error_cell` option.

Note:

The `read_drc_error_file` command supports only flat Calibre DRC error files; it does not support Calibre hierarchy DRC error files.

You can load the Milkyway error view created by the `read_drc_error_file` command into the error browser to report or display the DRC violations. For information about using the error browser, see “[Analyzing DRC Violations](#)” on page 6-133.

Analyzing DRC Violations

After you have generated a Milkyway error view, either by running one of the Synopsys design rule checking commands or by converting the Calibre results to an error view, you can analyze the DRC violations by

- Using the DRC query commands
- Using the error browser

The following sections describe these capabilities.

Using the DRC Query Commands

You can get information about DRC violations by using the `get_drc_errors` command to create a collection of DRC violations and then using the `get_attribute` command to query the attributes of the errors. Some attributes that provide information about DRC errors are `type`, `bbox`, `nets`, and `shapes`. Note that the availability of attribute values depends on the error type and the verification method used. For a list of all attributes associated with DRC errors, use the `list_attributes -application -class drc_error` command.

By default, the `get_drc_errors` command creates a collection that contains all DRC violations detected during the most recent run of the `verify_zrt_route` command. You can modify the query by

- Using the `-type` option to restrict the errors returned by the command to a specific error type. To determine the error type values, use the `list_drc_error_types` or `report_drc_error_type` commands.
- Using the `-bbox` option to restrict the query to a specific region of the design.
- Using the `-error_view` option to query a specific error view.

For example, to create a collection that contains all shorted nets detected by the most recent `verify_zrt_route` run, use the following command:

```
icc_shell> get_attribute [get_drc_errors -type {Short}] nets
```

To use the DRC query commands on a nondefault error view, such as from a third-party DRC error file, you must first open the error view, either by loading it into the error browser or by using the `open_mw_cel -not_as_current` command. For example,

```
icc_shell> read_drc_error_file -error_cell Route_Opt.err error.file
icc_shell> set cellId [open_mw_cel -not_as_current Route_Opt.err]
icc_shell> report_drc_error_type -error_view $cellId
icc_shell> get_drc_errors -error_view Route_Opt.err
```

Using the Error Browser

The error browser lets you examine routing design rule errors in the GUI. You can filter the error list, highlight errors in the layout view, display information about an error on the Query panel, select errors interactively with the Error Selection tool, mark errors as fixed, and save the errors in a file.

For more information about using the error browser, see

- The Error Browser usage help

To view the Error Browser usage help, choose Help > Error Browser in the error browser window.

- The “Examining Routing and Verification Errors” topic in IC Compiler Help

To view IC Compiler Help, choose Help > IC Compiler Online Help in the GUI.

- The “[Examining Routing and Verification Errors](#)” section on [page A-94](#)

Routing Nets and Buses in the GUI

The layout editor in the IC Compiler GUI provides the following routing capabilities:

- [Routing Single Nets](#)
- [Creating Physical Buses](#)
- [Modifying Buses](#)
- [Routing Buses or Multiple Nets](#)
- [Creating Custom Wires](#)

The following sections describe these capabilities.

Note:

When you route nets and buses in the GUI, you can use the editor DRC capability to verify the routing results. For information about the editor DRC capability, see “[Viewing and Fixing Errors](#)” on [page A-96](#).

Routing Single Nets

You can route nets interactively in the active layout view by drawing the route segments with the Create Route tool. To activate the Create Route tool, click the  button on the Edit toolbar or choose Edit > Create > Route Tool. A route can be composed of horizontal and vertical segments. As you draw the route segments, the tool guides you by displaying flylines and target port and pin locations. It can also check for DRC violations and display the estimated resistance of the segments. You can draw individual segments or dual orthogonal segments. By default, the tool snaps the route segments to the routing track edges.

Note:

The Create Route tool routes a single net. If you need to route multiple nets or physical buses, use the Advanced Route tool. For more information, see “[Routing Buses or Multiple Nets](#)” on [page 6-140](#).

After enabling the Create Route tool, you start a route by selecting a net. The default method for selecting a net is to click a physical object with a net connection in the layout view. You can click a net shape, user shape, pin, port, terminal, or via. The Create Route tool automatically sets the net and the routing layer based on the object you click. This capability is called automatic first-click selection. You can disable this capability and explicitly select the net and routing layer.

If the selected net uses a nondefault routing rule, by default, the Create Route tool honors the metal width requirement from the nondefault routing rule. If the selected net does not use a nondefault routing rule or you choose not to honor the nondefault routing rule, the Create Route tool uses the metal width requirement that is defined in the technology file.

To draw route segments, you click points in the layout view. The tool displays flylines and target port and pin locations to guide you in drawing the route segments. You can set options to adjust the routing, control the tool operation, and enable or disable routing aids.

To check for DRC violations as you draw the route segments, you must enable editor DRC by selecting the DRC option on the Mouse Tool Options toolbar or in the Create Route Tool dialog box. When you enable editor DRC, it checks for DRC violations as you draw based on the DRC rules that you set in the Editor DRC Options dialog box. When DRC violations occur, it displays visual cues to alert you to the violation, but does not prevent you from making the changes. If you do not enable editor DRC, you can check for DRC violations after you finish routing the net by choosing Edit > Editor DRC > Run DRC or Edit > Editor DRC > Run DRC on Selected Nets or by running the `gui_check_drc_errors` command. For more information about using editor DRC, see “[Viewing and Fixing Errors](#)” on page A-96.

To display the estimated resistance values, enable resistance estimation by selecting the Resistance option on the Mouse Tool Options toolbar or in the Create Route Tool dialog box. To adjust the criteria for resistance estimation, click the Options button beside the Resistance option in the Create Route Tool Options dialog box and set options as needed in the Set Resistance per Layer dialog box. For more information about resistance estimation, see the “[Viewing Estimated Wire Resistance](#)” topic in IC Compiler Help.

You can reverse and reapply Create Route tool operations by using the GUI undo and redo capabilities.

If you need assistance while using the Create Route tool, click  to open a Help page in the man page viewer.

For more information about using the Create Route tool, see the “[Routing Single Nets](#)” topic in IC Compiler Help. To view IC Compiler Help, choose Help > IC Compiler Online Help in the GUI.

You can also create routes from the command line by using the `create_net_shape` command with the `-type wire` option.

Creating Physical Buses

The layout editor can create physical buses in the following ways:

- Manual bus creation

To manually create a physical bus, you specify and sort the nets that you want to include in the bus.

- Automatic bus creation

To have the layout editor automatically create a physical bus, you define a net name pattern, and then the tool constructs the bus from nets with matching names.

Manual Bus Creation

You can manually create a physical bus by specifying and sorting the nets that you want to include in the bus. You can specify the nets by selecting them, searching for them by name, or loading their names from a file. You can also set net order and precedence options and select the index delimiter.

To manually create a physical bus,

1. Choose Edit > Create > Physical Buses.

The Physical Buses dialog box appears. If your design already contains physical buses, the bus names appear in the “Bus name” list.

2. Click the Create button.

The Create Physical Bus dialog box appears.

3. Type a unique bus name in the “Bus name” box.

4. Set the net insertion and sorting options as needed.

5. Specify the nets that you want to include in the bus.

You can search for the nets by name, insert selected nets, or load the nets from a file.

The names of the specified nets appear in the “Net name” list.

6. (Optional) Sort or remove nets in the “Net name: list as needed.

7. Click OK or Apply.

The tool displays the bus name in the Physical Buses dialog box.

Alternatively, you can use the `create_physical_bus` command.

For more information about manually creating a physical bus, see the “Creating Buses” topic in IC Compiler Help. To view IC Compiler Help, choose Help > IC Compiler Online Help in the GUI.

Automatic Bus Creation

You can automatically create a physical bus by defining a net name pattern and letting the tool construct the bus from nets with matching names. You define the net name pattern by specifying a name prefix and selecting one or more index delimiters. You can also set the index order, ascending or descending, and bus size constraint options.

To automatically create a physical bus,

1. Choose Edit > Create > Physical Buses.

The Physical Buses dialog box appears. If your design already contains physical buses, the bus names appear in the “Bus name” list.

2. Click the Auto Define button.

The Auto Define Buses dialog box appears.

3. Type a prefix in the “Net name prefix” box.

4. Select a net order option.

The choices are “ascending” and “descending.” The default is “ascending.”

5. Select or deselect bus index delimiter options under “Accept net name patterns” as needed to match the names of the nets that you want to include in the bus.

By default, all of the bus delimiter options are selected. To exclude nets with names that include specific delimiters, deselect the options for those delimiters. The available delimiters are braces ({i}), square brackets ([i]), angle brackets (<i>), underlines (_i_), colons (:i:), and parentheses ((i)).

6. Set the bus size constraint options as needed.

- To set the minimum number of nets in the bus, select or type a value in the Min box. The default is 8.
- To set the maximum number of nets in the bus, select or type a value in the Max box. The default is 128.
- To restrict the number of signals permitted in the bus, select or type a value in the “Multiple of” box. The default is 1.

7. Click OK or Apply.

Alternatively, you can use the `create_physical_buses_from_patterns` command.

For more information about automatically creating a physical bus, see the “Defining Buses Automatically” topic in IC Compiler Help. To view IC Compiler Help, choose Help > IC Compiler Online Help in the GUI.

Modifying Buses

You can modify a physical bus by specifying and sorting the nets that you want to include in the bus. You can specify the nets by selecting them, searching for them by name, or loading their names from a file. You can also set net order and precedence options and select the index delimiter.

To modify a physical bus,

1. Choose Edit > Create > Physical Buses.

The Physical Buses dialog box appears. The names of the existing buses appear in the “Bus name” list, and the names of their constituent nets appear in the “Net name” list.

2. Select the bus that you want to modify from the “Bus name” list.
3. Click the Modify button.

The Modify Physical Bus dialog box appears. The name of the selected bus appears in the “Bus name” box, and the names of its constituent nets appear in the “Net name” list.

4. Set net insertion and sorting options as needed.
 - If you want the tool to sort the nets automatically when you insert them, select the “Before selection” option.

By default, the “After selection” option is selected, which means the tool does not automatically sort the nets when you insert them.

- To set the net order, select a “Sort mode” option.

The choices are Ascending, Descending, and None. The default is Ascending.

- To set the net precedence, select a Precedence option.

The choices are Left and Right. The default is Left.

- To set the index delimiter character for the bus, select a “Bus delimiter” option.

The available delimiters are square brackets ([i]), angle brackets (<i>), braces ({i}), parentheses ((i)), underlines (_i_), and colons (:i:). The default is square brackets.

5. (Optional) Add nets to the bus as needed.

You can search for the nets by name, insert selected, or load the nets from a file.

- To search for the nets by name, click the Search button to open the Object Chooser dialog box, set the search parameters as needed, click the Search button, edit the “Search results” list as needed, and click OK.
- To insert selected nets, select the nets and click the Get Selection button. Alternatively, you can control the insertion order by clicking the arrow button beside the Get Selection button and choosing a command on the menu that appears.
- To load the nets from a file and preserve the net ordering specified in the file, click the “Load from file” button to open the “Read nets from file” dialog box, navigate to the directory where the file is located, select the file or type the file name, and click Open.

The names of the nets appear in the “Net name” list.

6. (Optional) Sort or remove nets in the “Net name” list as needed.

- To sort the nets, set “Sort mode” and precedence options as needed, and click the Sort button.
- To remove individual nets, select the nets and click the Remove button.
- To remove all of the nets, click the Remove All button.

7. Click OK or Apply.

The tool displays the bus name in the Physical Buses dialog box.

Alternatively, you can use the `create_physical_bus` command.

For more information about modifying a physical bus, see the “Modifying Buses” topic in IC Compiler Help. To view IC Compiler Help, choose Help > IC Compiler Online Help in the GUI.

Routing Buses or Multiple Nets

You can route a physical bus or multiple nets interactively in the active layout view by drawing the routes with the Advanced Route tool. To activate the Advanced Route tool, click the  button on the Edit toolbar or choose Edit > Create > Advanced Route Tool. Each route can be composed of horizontal and vertical segments. As you draw the route segments, the tool

guides you by displaying flylines and target port and pin locations. The tool can also interactively check for routing design rule violations as you draw. You can draw individual segments or dual orthogonal segments. By default, the tool snaps the route segments to the routing track edges.

Note:

The Advanced Route tool routes buses or multiple nets. If you need to route individual nets, use the Create Route tool. For more information, see “[Routing Single Nets](#)” on page 6-135.

After enabling the Advanced Route tool, you start a route by selecting a bus or net. The default method for selecting a bus or net is to click a physical object with a net connection in the layout view. You can click a net shape, user shape, pin, port, terminal, or via. The Create Route tool automatically sets the net and the routing layer based on the object you click. This capability is called automatic first-click selection. You can disable this capability and explicitly select the net and routing layer.

By default, the Advanced Route tool ignores routing blockages defined by the `create_routing_blockage` command. You can force the tool to honor these blockages by changing the setting in the Mouse Tool Options panel or the Advanced Route tool dialog box. The Advanced Route tool does not honor route guides defined by the `create_route_guide` command.

By default, if a selected net uses a nondefault routing rule, the Advanced Route tool honors the metal width and metal spacing requirements from the nondefault routing rule and uses these settings to determine the width and pitch of the routes. Note that the shielding width and shield spacing defined in the nondefault routing rule are not used by the Advanced Route tool. If a selected net does not use a nondefault routing rule or you choose not to honor the nondefault routing rules, the Advanced Route tool uses the metal width and metal spacing requirements defined in the technology file.

To draw route segments, you click points in the layout view. The tool displays flylines and target port and pin locations to guide you in drawing the route segments. It can also check for routing design rule violations as you draw the route segments. You can set options to adjust the routing, control the tool operation, and enable or disable routing aids.

You can reverse and reapply Advanced Route tool operations by using the GUI undo and redo capabilities.

By default, the Advanced Route tool displays error markers for any editor DRC violations that occur as you route the bus or nets. When editor DRC detects DRC violations, it automatically creates an editor DRC error view and loads it into the error browser. Editor DRC provides a limited set of advanced DRC rules that the advanced route editing tools can use to provide immediate visual feedback when a violation occurs during interactive routing. The tools support interactive DRC for advanced technologies as defined in the technology file. You can disable editor DRC or checks for individual DRC rules. If you disable editor DRC, you can check the bus or nets that you routed by choosing Edit > Editor DRC > Run

DRC or Edit > Editor DRC > Run DRC on Selected Nets or by running the `gui_check_drc_errors` command.

If you need assistance while using the Advanced Route tool, click  to open a Help page in the man page viewer.

For more information about using the Advanced Route tool, see the “Routing Buses or Multiple Nets” topic in IC Compiler Help. To view IC Compiler Help, choose Help > IC Compiler Online Help in the GUI. For more information about using editor DRC, see “[Viewing and Fixing Errors](#)” on page A-96.

Modifying Routed Nets

You can modify routed nets in the GUI by using the following tools:

- Area Push tool

To activate the Area Push tool, click the  button on the Edit toolbar or choose Edit > Area Push. You can use the Area Push tool to move unfixed objects away from a rectangular area on a layer while maintaining their physical connections. You select the layer and control whether the tool complies with nondefault routing rules. The tool supports both interactive and batch push operations.

If you need assistance while using the Area Push tool, click  to open a Help page in the man page viewer.

For more information about using the Area Push tool, see the “Pushing Objects” topic in IC Compiler Help. To view IC Compiler Help, choose Help > IC Compiler Online Help in the GUI.

- Spread Wire tool

To activate the Spread Wire tool, click the  button on the Edit toolbar or choose Edit > Spread Wires. You can use the Spread Wire to move selected, unfixed wires evenly between two points on a layer while maintaining their physical connections. You control whether the tool spreads the wires by layer and whether the tool complies with nondefault routing rules.

If you need assistance while using the Spread Wire tool, click  to open a Help page in the man page viewer.

For more information about using the Area Push tool, see the “Spreading Wires” topic in IC Compiler Help. To view IC Compiler Help, choose Help > IC Compiler Online Help in the GUI.

- Stretch Wire tool

To activate the Stretch Wire tool, click the  button or choose Edit > Stretch Wire. You can use the Stretch Wire tool to move and stretch unfixed wire shapes while optionally maintaining their physical connections.

If you need assistance while using the Stretch Wire tool, click  to open a Help page in the man page viewer.

For more information about using the Area Push tool, see the “Stretching Wires” topic in IC Compiler Help. To view IC Compiler Help, choose Help > IC Compiler Online Help in the GUI.

- Window Stretch tool

To activate the Window Stretch tool, click the  button or choose Edit > Window Stretch. You can use the Window Stretch tool to move or stretch unfixed objects within a rectangular area while optionally maintaining their physical connections. The tool moves all movable objects that are entirely inside the area and stretches all resizable objects that are partially inside the area.

If you need assistance while using the Window Stretch tool, click  to open a Help page in the man page viewer.

For more information about using the Area Push tool, see the “Moving and Stretching Objects” topic in IC Compiler Help. To view IC Compiler Help, choose Help > IC Compiler Online Help in the GUI.

The Area Push tool and the Spread Wire tool do not create DRC violations, but might fix existing DRC violations. To check for DRC violations after using these tools, run editor DRC by choosing Edit > Editor DRC > Run DRC or Edit > Editor DRC > Run DRC on Selected Nets or by running the `gui_check_drc_errors` command.

By default, the Stretch Wire tool and the Window Stretch tool display error markers for any editor DRC violations that occur as you route the bus or nets. When editor DRC detects DRC violations, it automatically creates an editor DRC error view and loads it into the error browser. Editor DRC provides a limited set of advanced DRC rules that the advanced route editing tools can use to provide immediate visual feedback when a violation occurs during interactive routing. The tools support interactive DRC for advanced technologies as defined in the technology file. You can disable editor DRC or checks for individual DRC rules. If you disable editor DRC, you can check the bus or nets that you routed by choosing Edit > Editor DRC > Run DRC or Edit > Editor DRC > Run DRC on Selected Nets or by running the `gui_check_drc_errors` command.

For more information about using editor DRC, see “[Viewing and Fixing Errors](#)” on page [A-96](#).

Creating Custom Wires

You can route one or more nets between specific points by using the Custom Wire tool. You specify the points by clicking in the active layout view and define the routes by setting options in the Create Custom Wires dialog box. A route can be composed of one or more horizontal or vertical segments. Each point you click after the first point specifies a route segment.

To route nets with the Custom Wire tool,

1. (Optional) Select a net shape, pin, port, terminal, user shape, or via to identify the net you need to route. If you perform this step, the tool operates in preselection mode.
2. Click the Custom Wire tool icon () in the Edit toolbar or choose Edit > Create > Custom Wires.

The Create Custom Wires dialog box appears. If you preselected the net, the GUI automatically sets the net name, layer, and width options.

3. (Optional) Set options as needed in the Create Custom Wires dialog box.

If you did not select the nets that you need to route before activating the Custom Wire tool, you can select or browse for the nets or type the net names in the Net box.

If you need to reset the Create Custom Wires dialog box options to their defaults, click Default. For details about setting options in the Create Custom Wires dialog box, see the “Setting Route Tool Options” topic in IC Compiler Help. To view IC Compiler Help, choose Help > IC Compiler Online Help in the GUI.

4. Click the startpoint for the routes in the active layout view.

If you selected the automatic first-click Bus or Net, Layer, or Width options and the object you click is a net shape, user shape, pin, port, terminal, or via, the tool automatically identifies the net connected to the object and can perform the following actions:

- Select the net and update the Net box in the Create Custom Wires dialog box.
- Identify the preferred routing layer for the net and update the Layer list in the Create Custom Wires dialog box.
- Identify the default wire width for the net and update the Width box in the Create Custom Wires dialog box.

The Net, Layer, and Width options are disabled by default.

5. Draw the first wire segments.

As you move the pointer, a ghost image appears between the initial point and the current pointer position. The color of the ghost image indicates the layer on which you are drawing the routes.

If you are routing multiple nets, the default reference wire is the low end segment, the bottom segment for horizontal routes or the leftmost segment for vertical routes, in the first section of the routes. If you want to use the high end segment instead, select the “High end” option in the Create Custom Wires dialog box.

6. Click a point or press Return when you need to finish the route segments and change the routing direction or the layer.

- To change the layer, press F1 or Shift-F1 repeatedly until the ghost shape color matches the color of the layer you want to use. To move up one layer, press F1. To move down one layer, press Shift-F1. When you change the layer, the tool automatically inserts the appropriate vias.
- To remove the last point, press Backspace or right-click and choose Remove Last Point. You can remove multiple points sequentially.
- To remove the entire route and start over, press Escape or right-click and choose Cancel.

If necessary, you can adjust options in the Create Custom Wires dialog box to control how the routes change direction.

- To switch the reference wire from one end to the other, select the “Switch ends” option.
- To change the number wires that each wire crosses, select or type a value in the “Skip count” box.

7. Repeat steps 5 and 6 as needed.
8. Finish the routes and instantiate the routed nets by doing one of the following:
 - Double-click a target pin or port.
 - Right-click and choose Apply.
 - Click Apply in the Create Custom Wires dialog box.
9. (Optional) Do one of the following:
 - To route additional nets, repeat steps 3 through 9.
 - To deactivate the Custom Wires Tool, press the Esc key or click OK in the Create Custom Wires dialog box.

For more information about creating custom wires, see the “Creating Custom Wires” topic in IC Compiler Help. To view IC Compiler Help, choose Help > IC Compiler Online Help in the GUI.

Postroute RC Extraction

The IC Compiler tool automatically performs postroute RC estimation when you run the `route_opt` command and when you run any of the following timing analysis commands on a routed but not extracted design: `report_timing`, `report_qor`, `report_constraint`, `report_delay_calculation`, `report_net`, `report_clock`, `report_clock_timing`, or `report_clock_tree`. In addition, you can explicitly perform postroute RC extraction by running the `extract_rc` command.

Note:

If the extraction is already done by a timing or extraction command, it does not occur again for the same routed database when you use the reporting commands.

By default, the `extract_rc` command performs RC estimation on any unrouted nets in your design and performs RC extraction on the routed nets in your design. If your design contains a mix of routed and unrouted nets, and you do not want to run RC estimation on the unrouted nets, use the `extract_rc -routed_nets_only` command, which performs only RC extraction.

When there are only a small number of routing changes, such as in the ECO flow, you can use the `extract_rc -incremental` command to perform incremental extraction of only the changed nets. Incremental extraction is performed based on the previous extraction in the same session.

During postroute RC extraction, the tool performs the following tasks:

- Determines the RC coefficients

The tool derives the RC coefficients from the TLUPlus models (for information about attaching the TLUPlus data to your Milkyway design library, see “[Setting Up the TLUPlus Files](#)” on page 3-40).

- Calculates the RC values

If your design contains global routing, the tool uses a global route extraction model to calculate the parasitics. This model uses resistance values derived from the global routing and estimated capacitance values.

If your design contains track assignments or detail routing, the tool uses a detail route extraction model to calculate the parasitics. This model uses resistance and capacitance values derived from the detail postroute geometries.

The tool does not extract the coupling capacitance values unless you have set the delta delay signal integrity option to `true` (`set_si_options -delta_delay true`). To force extraction of these values, run the `extract_rc -coupling_cap` command.

- Saves the timing data and attributes in the Milkyway design library
The parasitics are saved in the Milkyway design library, but a parasitic data file is not saved. To save the parasitics in a data file, run the `write_parasitics` command.

To fine-tune the postroute RC extraction, you specify the scaling factors by using the `set_extraction_options` command and its options, as shown in [Table 6-11](#).

Table 6-11 set_extraction_options Command Options

To do this	Use this
Scale the capacitance coefficients.	<code>-max_cap_scale</code> <code>-min_cap_scale</code>
Scale the resistance coefficients.	<code>-max_res_scale</code> <code>-min_res_scale</code>
Scale the coupling capacitance coefficients.	<code>-max_ccap_scale</code> <code>-min_ccap_scale</code>
Specify the coupling capacitance net-to-net filtering threshold (default is 0.003 picofarad).	<code>-max_net_ccap_threshold</code> <code>-min_net_ccap_threshold</code>
Specify the coupling capacitance net-to-net filtering ratio (default is 0.03 of total capacitance).	<code>-max_net_ccap_ratio</code> <code>-min_net_ccap_ratio</code>
Specify the coupling capacitance average net-to-net filtering ratio (default is 0.25 of average coupling capacitance).	<code>-max_net_ccap_avg_ratio</code> <code>-min_net_ccap_avg_ratio</code>
Specify the process scaling factors (default is 1.0).	<code>-max_process_scale</code> <code>-min_process_scale</code>
Disable the breaking of long wire segments during extraction.	<code>-no_break_segments</code>
Specify the maximum length of wire segments (default is 50 microns). If you use <code>-no_break_segments</code> , this option has no effect.	<code>-max_segment_length</code>
Specify the type of real metal fill extraction to perform. The default is none.	<code>-real_metalfill_extraction</code> <code>none floating grounded auto</code>
Control whether virtual shield extraction is performed. The default is <code>true</code> .	<code>-virtual_shield_extraction</code> <code>true false</code>

Table 6-11 set_extraction_options Command Options (Continued)

To do this	Use this
Specify the fanout threshold. The default is 1000 pins.	<code>-fan_out_threshold count</code>
Specify the pattern density outside the block for extraction. The default is 0.0.	<code>-density_outside_block density_value</code>
Specify the reference direction to use when using data from the orientation-based etch tables in the ITF file. The default is <code>use_from_tluplus</code> .	<code>-reference_direction vertical horizontal use_from_tluplus</code>
Control whether a detailed resistance network is created for pins with multiple connections. The default is <code>false</code> .	<code>-include_pin_resistance true false</code>
Reset extraction options.	<code>-default</code>

For multicorner-multimode designs, use the `set_extraction_options` command to set the resistance, capacitance, and coupling capacitance scaling values for the current scenario. For information about setting the current scenario see “[Defining Scenarios](#)” on page 3-34. The following options of the `set_extraction_options` command can be specified for the current scenario:

- `-max_res_scale`
- `-min_res_scale`
- `-max_cap_scale`
- `-min_cap_scale`
- `-max_ccap_scale`
- `-min_ccap_scale`

Note:

The other options supported by the `set_extraction_options` command are not scenario-specific.

To check the values of the specified scaling factors, use the `report_extraction_options` command.

You can use process scaling factors to adjust RC estimation and extraction for a process shrink, such as shrinking from 90 nm to 65 nm. The lengths and widths of the interconnect wires are scaled by the specified factors, without affecting interconnect wire depths or metal geometries inside standard cells and macros. You specify the scaling factors with the

`-max_process_scale` and `-min_process_scale` options of the `set_extraction_options` command. These factors affect the RC values generated by the `extract_rc` command for maximum-delay and minimum-delay analysis, respectively, for both preroute RC estimation and postroute RC extraction.

Note:

The process scaling factors of the `set_extraction_options` command are the only settings that apply to both preroute RC estimation and postroute RC extraction. All other options of the `set_extraction_options` command apply only to postroute RC extraction.

7

Chip Finishing and Design for Manufacturing

The IC Compiler tool provides chip finishing and design for manufacturing and yield capabilities that you can apply throughout the various stages of the design flow to address process design issues encountered during chip manufacturing.

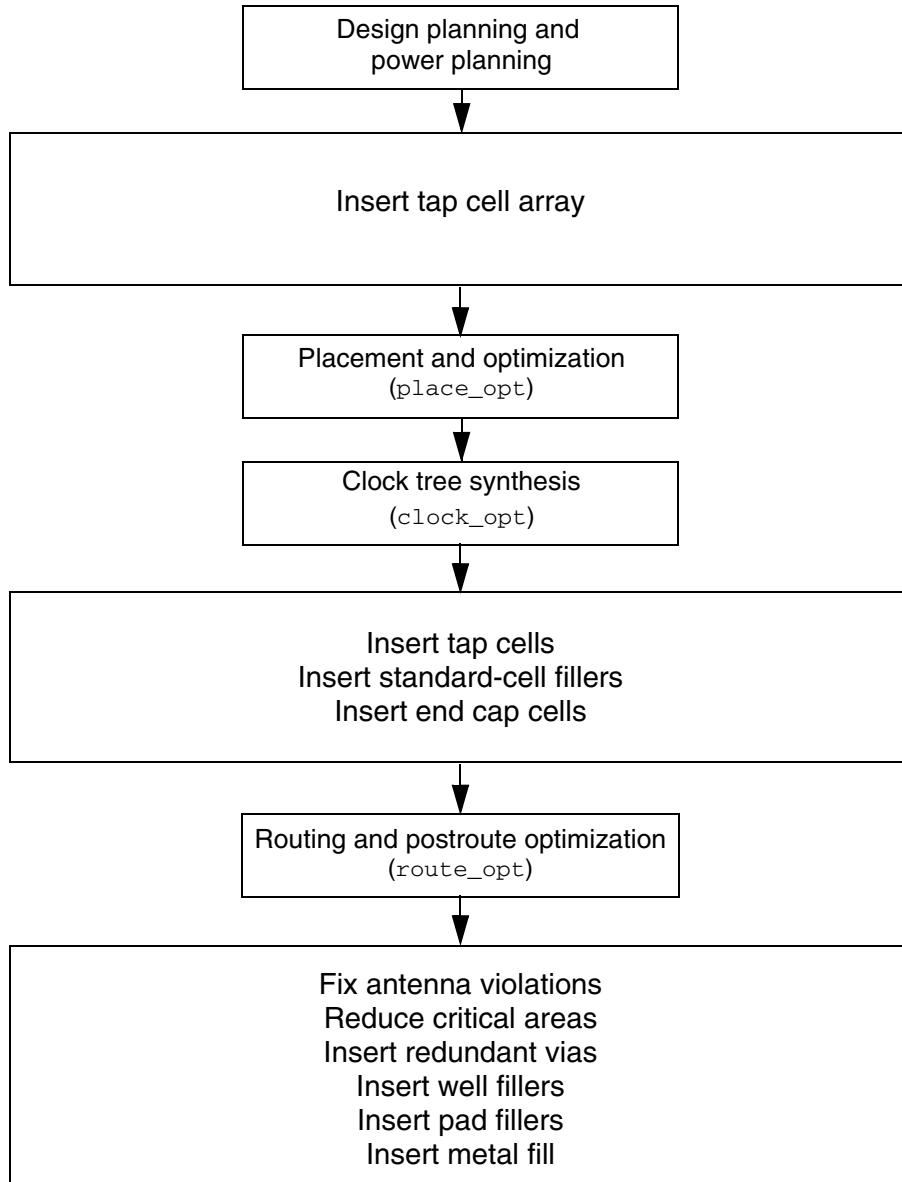
This chapter contains the following sections:

- [Overview](#)
- [Inserting Tap Cells](#)
- [Finding and Fixing Antenna Violations](#)
- [Inserting Redundant Vias](#)
- [Optimizing Wire Length and Via Count](#)
- [Reducing Critical Areas](#)
- [Shielding Nets](#)
- [Inserting Filler Cells](#)
- [Inserting Metal Fill](#)

Overview

[Figure 7-1](#) shows the design for manufacturing and chip finishing tasks supported by the IC Compiler tool and how these tasks fit into the overall IC Compiler design flow. The following sections describe how to perform these tasks.

Figure 7-1 Design for Manufacturing and Chip Finishing Tasks in the Design Flow



Inserting Tap Cells

Tap cells are a special nonlogic cell with well and substrate ties. These cells are typically used when most or all of the standard cells in the library contain no substrate or well taps. Generally, the design rules specify the maximum distance allowed between every transistor in a standard cell and a well or the substrate ties.

You can insert tap cells in your design before or after placement:

- You can insert tap cell arrays before placement to ensure that the placement complies with the maximum diffusion-to-tap limit.
- You can insert tap cells after placement to fix maximum diffusion-to-tap violations.

The following sections describe these methods.

Adding Tap Cell Arrays

Before global placement (during the floorplanning stage), you can add tap cells to the design that form a two-dimensional array structure to ensure that all standard cells placed subsequently comply with the maximum diffusion-to-tap distance limit.

You need to specify the tap distance and offset, based on your specific design rule distance limit. The command has no knowledge of the design rule distance limit. After you run the command, visually check to ensure that all standard-cell placeable areas are properly protected by tap cells.

To add a tap cell array, use the `add_tap_cell_array` command (or choose Finishing > Add Tap Cell Array in the GUI).

For example,

```
icc_shell> add_tap_cell_array -master_cell_name Cell1 -distance 30 \
    -pattern normal -voltage_area [get_voltage_areas "V*"] \
    -no_tap_cell_under_layers {M1 M2}
```

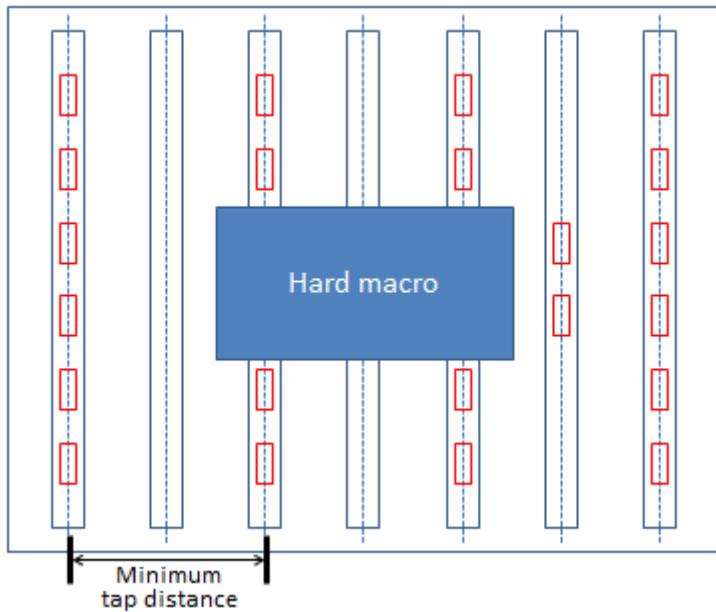
You must use the `-master_cell_name` option to specify the name of the reference cell to be used for tap cell insertion and the `-distance` option to specify the required distance, in microns, between tap cells. The specified distance should be approximately twice the maximum diffusion-to-tap value specified in the technology design rule. Use the `-voltage_area` option to restrict tap insertion to specific voltage areas. Use the `-plan_group` option to restrict tap insertion to specific plan groups.

You specify the insertion pattern by using the `-pattern` option: `normal` (the default), `every_other_row`, or `stagger_every_other_row`. “Normal” insertion adds tap cells to every row, using the specified distance limit. “Every other row” insertion adds tap cells in the odd-numbered rows only, which reduces the added tap cells by one-half. “Stagger every

“other row” insertion adds tap cells in every row with tap cells in even rows offset by half the specified `-offset` setting relative to the odd rows, producing a checkerboard-like pattern.

The minimum tap distance is half the value specified by the `-distance` option. By default, if a tap cell does not exist within this distance from each row edge adjacent to the design’s boundary, a hard macro, or a placement blockage, the tool inserts an additional tap cell. [Figure 7-2](#) shows the default tap cell placement for a design that uses the normal insertion pattern. Note that extra tap cells are added to the right of the hard macro to ensure that a tap cell exists within the minimum tap distance from the edge of the hard macro.

Figure 7-2 Default Tap Cell Placement



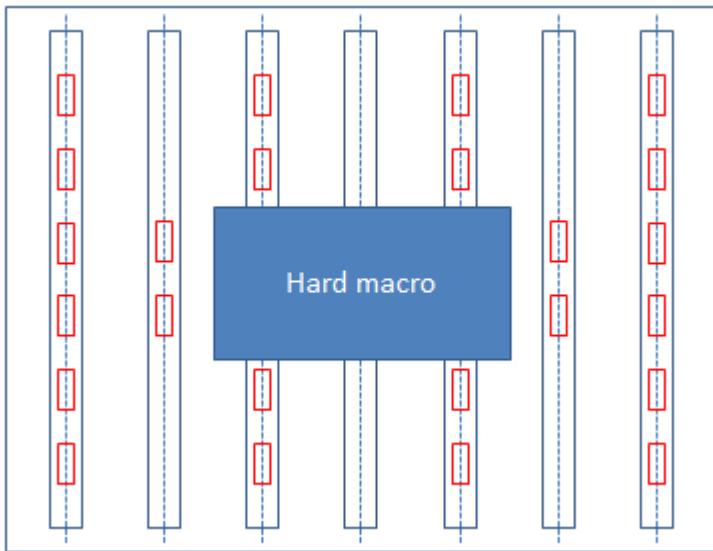
You can control the insertion of tap cells adjacent to the design’s boundary, hard macros, and placement blockages by using the following options:

- `-left_boundary_extra_tap`
- `-right_boundary_extra_tap`
- `-left_macro_blockage_extra_tap`
- `-right_macro_blockage_extra_tap`

To insert a tap cell when needed, specify `by_rule`, which is the default. To always insert tap cells, even if one already exists within this distance, specify `must_insert`. To never insert tap cells, specify `no_insert`.

For example, to require tap cells to the left of the hard macro in [Figure 7-2](#), use the `-left_boundary_extra_tap must_insert` option when you run the `add_tap_cell_array` command. [Figure 7-3](#) shows the resulting tap cell placement.

Figure 7-3 Must-Insert Tap Cell Placement



When you specify the `must_insert` keyword, the tool might insert more tap cells than are strictly required. To remove any redundant tap cells, use the `-remove_redundant_tap_cells` option. This option removes all redundant tap cells, regardless of whether they were inserted by the same command or were inserted earlier in the design process.

The command also has options to specify the tap cell naming conventions, well port and net names, substrate port and net names, ground net connections, and prohibited placement under specified metal layer preroutes.

For designs that are at the 28-nm process node or less, the foundry spacing rules require no filler insertion of one unit tile. However, by default, the `add_tap_cell_array` command might place tap cells at locations where one-unit-tile gaps can occur. To prevent one-unit-tile violations between tap cells and fixed cells, macros, blockages, or the chip boundaries, set the `-no_1x` option. When you set this option, the command moves tap cells to the next legal locations to avoid creating one-unit-tile gaps. This option cannot be used when the `-skip_fixed_cells` option is set to `false`.

Fixing Tap Spacing Violations

You can add tap cells to comply with the diffusion-to-substrate or -well-contact maximum spacing design rule. After global placement (typically), you can insert tap cells “by rules” so that all existing standard cells comply with the maximum diffusion-to-tap distance limit.

To insert tap cells to satisfy to the diffusion-to-tap design rules, use the `insert_tap_cells_by_rules` command (or choose Finishing > Insert Tap Cells By Rules in the GUI).

For example, to insert a tap cell named MY_TOP into a placed design to satisfy the condition that the distance from a standard cell to a tap cell must be no more than 30 microns, with the restriction that tap cells cannot be inserted under layer metal1, use the following command:

```
icc_shell> insert_tap_cells_by_rules -tap_master "MY_TAP" \
    -tap_cell_insertion -tap_distance_based -move \
    -tap_distance_limit 30.0 \
    -no_tap_cells_under_metal_layer {metal1}
```

You must use the `-tap_master` option to specify the name of the tap cell. In addition, you must use either the `-drc_spacing_check` or `-tap_cell_insertion` option to specify whether to perform only DRC tap distance checking or to perform both checking and fixing of violations by inserting tap cells.

You must use either the `-tap_distance_based` or `-drc_spacing_based` option to specify the method of calculating distances between standard cells and tap cells. When you use the `-tap_distance_based` option, the tool uses a simple measurement from the standard cell to the tap, irrespective of the actual metal and contact layers within the cells. You should use this method when the actual layout of the standard cells or tap cells is not available in the cell library. You must specify the tap distance limit by using the `-tap_distance_limit` option.

When you use the `-drc_spacing_based` option, the tool uses the actual layout of the standard cell and tap cell, with consideration of the diffusion, well, and contact layers in the cells. This method results in the insertion of fewer tap cells because the tool recognizes existing taps in the standard cells. You must specify the applicable layer names and design rule distance using options such as `-n_well_layer` and `-tap_spacing_design_rule`.

You must use either the `-move` or `-freeze` option to specify whether to allow standard cells to be moved to make room for tap cells. Allowing standard cells to move can help prevent design rule violations and reduce the number of tap cells that must be inserted but can affect the design timing.

The command also has options to specify the tap layer name, tap cell naming conventions, respect for hard and soft blockages, allowance of tap sharing, power and ground net connections, prohibited placement under specified metal layer preroutes, voltage areas in which to perform tap cell insertion, and respect for hard macro keepout margins.

Removing Tap Cells

To remove tap cells, use the `remove_stdcell_filler -tap` command or choose Finishing > Remove Fillers in the GUI and select Tap as the filler type to remove. You can optionally specify a bounding box from which to remove the tap cells.

Finding and Fixing Antenna Violations

In chip manufacturing, gate oxide can be easily damaged by electrostatic discharge. The static charge that is collected on wires during the multilevel metallization process can damage the device or lead to a total chip failure. The phenomenon of an electrostatic charge being discharged into the device is referred to as either antenna or charge-collecting antenna problems.

To prevent antenna problems, the tool verifies that for each input pin the metal antenna area divided by the gate area is less than the maximum antenna ratio given by the foundry:

$$\text{(antenna-area)/(gate-area)} < \text{(max-antenna-ratio)}$$

The antenna flow consists of the following steps:

1. Define the antenna rules
2. Specify the antenna properties of the pins and ports
3. Analyze and fix the antenna violations

The following sections describe these steps:

- [Defining Metal Layer Antenna Rules](#)
- [Specifying Antenna Properties](#)
- [Analyzing and Fixing Antenna Violations](#)

Defining Metal Layer Antenna Rules

You define the metal layer antenna rules by defining both global metal layer antenna rules and layer-specific metal layer antenna rules. You define the global metal layer antenna rules by using the `define_antenna_rule` command. You define the layer-specific antenna rules by using the `define_antenna_layer_rule` command.

The following commands show an example of defining the metal layer antenna rules:

```
set lib [current_mw_lib]
remove_antenna_rules $lib
define_antenna_rule $lib -mode 1 -diode_mode 4 \
    -metal_ratio 300 -cut_ratio 20
define_antenna_layer_rule $lib -mode 1 -layer "M1" \
    -ratio 300 -diode_ratio {0.09 0 123 16880}
define_antenna_layer_rule $lib -mode 1 -layer "M2" \
    -ratio 300 -diode_ratio {0.09 0 123 16880}
define_antenna_layer_rule $lib -mode 1 -layer "M3" \
    -ratio 300 -diode_ratio {0.09 0 123 16880}
define_antenna_layer_rule $lib -mode 1 -layer "M4" \
    -ratio 300 -diode_ratio {0.09 0 123 16880}
define_antenna_layer_rule $lib -mode 1 -layer "M5" \
    -ratio 400 -diode_ratio {0.09 0 123 20000}
define_antenna_layer_rule $lib -mode 1 -layer "VIA1" \
    -ratio 20 -diode_ratio {0.09 0 110 500}
define_antenna_layer_rule $lib -mode 1 -layer "VIA2" \
    -ratio 20 -diode_ratio {0.09 0 110 500}
define_antenna_layer_rule $lib -mode 1 -layer "VIA3" \
    -ratio 20 -diode_ratio {0.09 0 110 500}
define_antenna_layer_rule $lib -mode 1 -layer "VIA4" \
    -ratio 20 -diode_ratio {0.09 0 110 500}
```

The following sections provide details about how to define the metal layer antenna rules.

Defining the Global Metal Layer Antenna Rules

You use the `define_antenna_rule` command to define the global metal layer antenna rules. These rules apply whenever a layer-specific antenna rule does not exist.

When you define the global metal layer antenna rules, you must specify

- The maximum antenna ratios

You must specify a maximum antenna ratio for metal layers and for via layers.

- To specify the maximum antenna ratio for metal layers, use the `-metal_ratio` option. This is a required option of the `define_antenna_rule` command.
- To specify the maximum antenna ratio for via layers, use the `-cut_ratio` option. This is a required option of the `define_antenna_rule` command.

- The way to calculate the antenna area

The antenna area calculation depends on which area mode to use and which metal segments to consider, which is referred to as the antenna recognition mode. You specify these settings by using the `-mode` option. This is a required option of the `define_antenna_rule` command. For information about setting the `-mode` option, see “[Setting the Antenna Mode](#)” on page 7-9.

- The diode protection mode

The IC Compiler tool supports thirteen diode protection modes. You must specify the diode protection mode by using the `-diode_mode` option. This is a required option of the `define_antenna_rule` command. For information about the diode protection modes, see “[Setting the Diode Protection Mode](#)” on page 7-13.

Setting the Antenna Mode

The IC Compiler tool supports six antenna modes. The mode value, as set by the `-mode` option, indicates which area calculation mode to use and which metal segments to consider. [Table 7-2 on page 7-12](#) shows the area calculation mode and the antenna recognition mode selected by each antenna mode value.

The tool supports the following area calculation modes:

- Surface area, which is calculated as $W \times L$
- Sidewall area, which is calculated as $(W + L) \times 2 \times \text{thickness}$

Note:

If you use sidewall area calculation, you must define the metal thickness by specifying the `unitMinThickness`, `unitNomThickness`, and `unitMaxThickness` attributes in each `Layer` section of the technology file.

The tool supports the following antenna recognition modes:

- Single-layer mode

In single-layer mode, the tool considers only the metal segments on the current layer; the metal segments on all lower layers are ignored. This mode allows the best routability. In this mode, the antenna ratio is calculated as

$$\text{antenna_ratio} = \frac{\text{connected metal area of the layer}}{\text{total gate area}}$$

- Accumulative ratio mode

In accumulative ratio mode, the tool considers the metal segments on the current layer and the lower-layer segments to the input pins. In this mode, the antenna ratio is calculated as

$$\text{antenna_ratio} = \text{accumulation of ratios for the layer and layers below}$$

- Accumulative area mode

In accumulative area mode, the tool considers the metal segments on the current layer and all lower-layer segments. In this mode, the antenna ratio is calculated as

$$\text{antenna_ratio} = \text{all connected metal areas} / \text{total gate area}$$

[Figure 7-4](#) shows a layout example with a lateral view, which is used to explain these antenna recognition modes. [Table 7-1](#) shows the antenna ratios for each antenna recognition mode for this layout example.

Figure 7-4 Layout Example

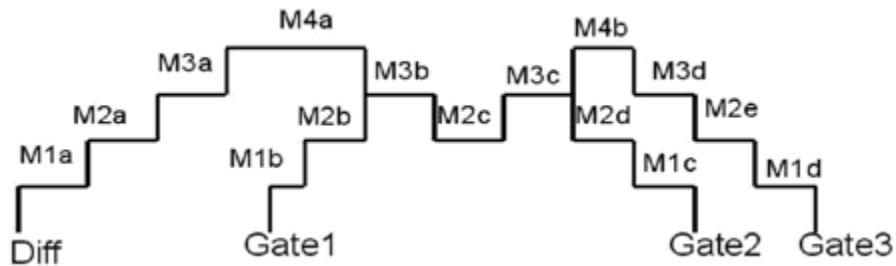


Table 7-1 Antenna Recognition Modes and Ratios

Considered segments	Antenna ratio
Single-layer mode	<p>M1 ratios</p> <ul style="list-style-type: none"> • Gate1: M1b / Gate1 • Gate2: M1c / Gate2 • Gate3: M1d / Gate3 <p>M2 ratios</p> <ul style="list-style-type: none"> • Gate1: M2b / Gate1 • Gate2: M2d / Gate2 • Gate3: M2e / Gate3 <p>M3 ratios</p> <ul style="list-style-type: none"> • Gate1, 2: $(M3b + M3c) / (Gate1 + Gate2)$ • Gate3: M3d / Gate3 <p>M4 ratios</p> <ul style="list-style-type: none"> • Gate1, 2, 3: $(M4a + M4b) / (Gate1 + Gate2 + Gate3)$
Accumulative ratio mode	<p>M1 ratios</p> <ul style="list-style-type: none"> • Gate1: M1b / Gate1 • Gate2: M1c / Gate2 • Gate3: M1d / Gate3 <p>M2 ratios</p> <ul style="list-style-type: none"> • Gate1: $M1b / (Gate1 + M2b / Gate1)$ • Gate2: $M1c / (Gate2 + M2d / Gate2)$ • Gate3: $M1d / (Gate3 + M2e / Gate3)$ <p>M3 ratios</p> <ul style="list-style-type: none"> • Gate1: $M1b / (Gate1 + M2b / Gate1 + (M3b + M3c) / (Gate1 + Gate2))$ • Gate2: $M1c / (Gate2 + M2d / Gate2 + (M3b + M3c) / (Gate1 + Gate2))$ • Gate3: $M1d / (Gate3 + M2e / Gate3 + M3d / Gate3)$ <p>M4 ratios</p> <ul style="list-style-type: none"> • Gate1: $M1b / (Gate1 + M2b / Gate1 + (M3b + M3c) / (Gate1 + Gate2) + (M4a + M4b) / (Gate1 + Gate2 + Gate3))$ • Gate2: $M1c / (Gate2 + M2d / Gate2 + (M3b + M3c) / (Gate1 + Gate2) + (M4a + M4b) / (Gate1 + Gate2 + Gate3))$ • Gate3: $M1d / (Gate3 + M2e / Gate3 + M3d / Gate3 + (M4a + M4b) / (Gate1 + Gate2 + Gate3))$

Table 7-1 Antenna Recognition Modes and Ratios (Continued)

Considered segments	Antenna ratio
Accumulative area mode	<p>M1 ratios</p> <ul style="list-style-type: none"> • Gate1: M1b / Gate1 • Gate2: M1c / Gate2 • Gate3: M1d / Gate3 <p>M2 ratios</p> <ul style="list-style-type: none"> • Gate1: (M1b + M2b) / Gate1 • Gate2: (M1c + M2d) / Gate2 • Gate3: (M1d + M2e) / Gate3 <p>M3 ratios</p> <ul style="list-style-type: none"> • Gate1, 2: (M1b + M1c + M2b + M2c + M2d + M3b + M3c) / (Gate1 + Gate2) • Gate3: (M1d + M2e + M3d) / Gate3 <p>M4 ratios</p> <ul style="list-style-type: none"> • Gate1, 2, 3: all metal areas / (Gate1 + Gate2 + Gate3)

[Table 7-2](#) shows the area calculation mode and the antenna recognition mode selected by each antenna mode value.

Table 7-2 Antenna Mode Settings

Antenna mode value	Area calculation mode	Antenna recognition mode
1	Surface	Single-layer
2	Surface	Accumulative ratio
3	Surface	Accumulative area
4	Sidewall	Single-layer
5	Sidewall	Accumulative ratio
6	Sidewall	Accumulative area

Setting the Diode Protection Mode

The diode protection mode specifies how much protection the diodes provide. Note that the tool considers all output pins to be diodes. [Table 7-3](#) defines each of the diode protection mode settings.

Table 7-3 Diode Protection Mode Settings

Diode protection mode	Definition
0	The diodes do not provide any protection.
1	The diodes provide unlimited protection. In this mode, the highest metal layer is not checked for antenna violations because the input-to-output connection is completed when the highest metal layer is formed.
2	Diode protection is limited; if more than one diode is connected, the largest value of the maximum antenna ratio for all diodes is used.
3	Diode protection is limited; if more than one diode is connected, the sum of the maximum antenna ratios for all diodes is used.
4	Diode protection is limited; if more than one diode is connected, the sum of the diode-protection values for all diodes is used to compute the maximum antenna ratio.
5	Diode protection is limited; the maximum diode-protection value for all diodes is used to calculate the equivalent gate area.
6	Diode protection is limited; the sum of the diode-protection values for all diodes is used to calculate the equivalent gate area.
7	Diode protection is limited; the maximum diode-protection value for all diodes is used to calculate the equivalent metal area.
8	Diode protection is limited; the sum of the diode-protection values for all diodes is used to calculate the equivalent metal area.
9	Diode protection is limited; scaling is based on maximum diode protection.
10	Diode protection is limited; scaling is based on total diode protection.

Table 7-3 Diode Protection Mode Settings (Continued)

Diode protection mode	Definition
11	Diode protection is limited; scaling is based on maximum diode protection.
12	Diode protection is limited; scaling is based on total diode protection.

Defining Layer-Specific Antenna Rules

You use the `define_antenna_layer_rule` command to define a layer-specific antenna rule. Layer-specific antenna rules override the global metal layer antenna rules for the specified layer.

When you define a layer-specific antenna rule, you must specify

- The layer

To specify the layer, use the `-layer` option. The layer can be either a metal layer or a via layer.

- The antenna mode

To specify the antenna mode, use the `-mode` option. The antenna mode should be the same as the one that was used in the `define_antenna_rule` command. For information about the antenna modes, see “[Setting the Antenna Mode](#)” on page 7-9.

- The diode ratios

The diode ratios specify how to perform antenna ratio calculation with diode protection. The calculation is based on the vector that you specify by using the `-diode_ratio` option. For information about specifying the diode ratio vector, see “[Specifying the Diode Ratio Vector](#)” on page 7-15.

Note:

If you do not specify a layer-specific antenna rule for a metal layer, the tool uses the global metal layer antenna rule set by the `define_antenna_rule` command and uses a diode ratio vector of {0 0 1 0} to perform antenna ratio calculation with diode protection. For more information about setting the global metal layer antenna rule, see “[Defining Metal Layer Antenna Rules](#)” on page 7-8. For more information about the meaning of the diode ratio vector, see “[Specifying the Diode Ratio Vector](#)” on page 7-15.

Specifying the Diode Ratio Vector

The diode ratio vector defines the maximum allowable antenna ratio (max-antenna-ratio) of the antenna area to the gate area if the antenna is protected by diodes. The antenna ratio of each metal layer must be less than the allowable max-antenna-ratio (antenna-area / gate-area < max-antenna-ratio).

The format of the diode ratio vector is

```
{v0 v1 v2 v3 [v4]}
```

The *v4* value represents the upper limit of the diode protection and is optional. If you do not specify the *v4* value, it is assumed to be 0, which means there is no upper limit.

Note:

Diode modes 11 and 12 use a second vector, {*s0 s1 s2 s3 s4 s5*} to specify scaling values. In this case, you specify the diode ratio vectors as

```
{}{v0 v1 v2 v3 [v4]}{s0 s1 s2 s3 s4 s5}{}{}
```

The actual usage of the diode ratio vector depends on the diode mode that was specified in the `define_antenna_rule` command. [Table 7-4](#) shows how the diode ratio vector is used for each of the diode modes. In this table, *dp* represents the diode protection value specified for an output pin. For information about specifying this value, see “[Specifying Antenna Properties](#)” on page [7-20](#).

Table 7-4 Antenna Ratio Calculation Based on Diode Mode

Diode mode	Calculation
0, 1	Not used
2, 3, 4	allowable max-antenna-ratio <ul style="list-style-type: none"> • If $dp > v0$ and $v4 <> 0$, allowable max-antenna-ratio = $\min(((dp + v1) * v2 + v3), v4)$ • If $dp > v0$ and $v4 = 0$, allowable max-antenna-ratio = $(dp + v1) * v2 + v3$ • If $dp \leq v0$, allowable max-antenna-ratio = layerMaxRatio
5	$\text{antenna_ratio} = \text{metal_area} / (\text{gate_area} + \text{equi_gate_area})$ <ul style="list-style-type: none"> • If $\text{max_diode_protection} > v0$ and $v4 <> 0$, $\text{equi_gate_area} = \min((\text{max_diode_protection} + v1) * v2 + v3), v4$ • If $\text{max_diode_protection} > v0$ and $v4 = 0$, $\text{equi_gate_area} = (\text{max_diode_protection} + v1) * v2 + v3$ • If $\text{max_diode_protection} \leq v0$, $\text{equi_gate_area} = 0$

Table 7-4 Antenna Ratio Calculation Based on Diode Mode (Continued)

Diode mode	Calculation
6	<pre>antenna_ratio = metal_area / (gate_area + equi_gate_area) • If total_diode_protection>v0 and v4<>0, equi_gate_area = min (((total_diode_protection + v1) * v2 + v3), v4) • If total_diode_protection>v0 and v4=0, equi_gate_area = (total_diode_protection + v1) * v2 + v3 • If total_diode_protection<=v0, equi_gate_area = 0</pre>
7	<pre>antenna_ratio = (metal_area - equi_metal_area) / gate_area • If max_diode_protection>v0 and v4<>0, equi_metal_area = min (((max_diode_protection + v1) * v2 + v3), v4) • If max_diode_protection>v0 and v4=0, equi_metal_area = (max_diode_protection + v1) * v2 + v3 • If max_diode_protection<=v0, equi_metal_area = 0 • If equi_metal_area>metal_area, equi_metal_area = metal_area</pre>
8	<pre>antenna_ratio = (metal_area - equi_metal_area) / gate_area • If total_diode_protection>v0 and v4<>0, equi_metal_area = min (((total_diode_protection + v1) * v2 + v3), v4) • If total_diode_protection>v0 and v4=0, equi_metal_area = (total_diode_protection + v1) * v2 + v3 • If total_diode_protection<=v0, equi_metal_area = 0 • If equi_metal_area > metal_area, equi_metal_area = metal_area</pre>
9	<pre>antenna_ratio = scale * metal_area / gate_area • If max_diode_protection>v0, scale = max (1 / ((max_diode_protection + v1) * v2 + v3), v4) • If max_diode_protection<=v0, scale = 1.0</pre>

Table 7-4 Antenna Ratio Calculation Based on Diode Mode (Continued)

Diode mode	Calculation
10	<pre>antenna_ratio = scale * metal_area / gate_area • If total_diode_protection>v0, scale = max (1 / ((total_diode_protection + v1) * v2 + v3), v4) • If total_diode_protection<= v0, scale = 1.0</pre>
11	<pre>antenna_ratio = scale * metal_area / gate_area • If max_diode_protection<v0, scale = s0 • If max_diode_protection<v1, scale = s1 • If max_diode_protection<v2, scale = s2 • If max_diode_protection<v3, scale = s3 • If max_diode_protection<v4, scale = s4 • If max_diode_protection>=v4, scale = s5</pre>
12	<pre>antenna_ratio = scale * metal_area / gate_area • If total_diode_protection<v0, scale = s0 • If total_diode_protection<v1, scale = s1 • If total_diode_protection<v2, scale = s2 • If total_diode_protection<v3, scale = s3 • If total_diode_protection<v4, scale = s4 • If total_diode_protection>=v4, scale = s5</pre>

Example for Diode Modes 2, 3, and 4

Assume that the diode ratio vector is {0.7 0.0 200 2000}; the layerMaxRatio value is 400; and the following diodes are connected to a single net: diode A with a diode-protection value of 0.5, diode B with a diode-protection value of 1.0, and diode C with a diode-protection value of 1.5.

The maximum antenna ratio for each diode is computed by using the diode ratio vector and the formula for diode modes 2 through 4:

- Diode A has a maximum antenna ratio of 400

The diode protection value of 0.5 is less than the $v0$ value of 0.7; therefore, the maximum antenna ratio is the layerMaxRatio value.

- Diode B has a maximum antenna ratio of 2200

The diode protection value of 1.0 is greater than the $v0$ value of 0.7, and $v4$ is equal to 0; therefore, the maximum antenna ratio is $(1.0+0.0) * 200 + 2000 = 2200$.

- Diode C has a maximum antenna ratio of 2300

The diode protection value of 1.5 is greater than the $v0$ value of 0.7, and $v4$ is equal to 0; therefore, the maximum antenna ratio is $(1.5+0.0) * 200 + 2000 = 2300$.

The maximum antenna ratio for the net is computed by using the formula for the diode mode:

- For diode mode 2, the maximum antenna ratio for the net is the largest of the maximum antenna ratio values for the diodes, 2300.
- For diode mode 3, the maximum antenna ratio for the net is the sum of the maximum antenna ratios for the diodes, $400 + 2200 + 2300 = 4900$.
- For diode mode 4, the maximum antenna ratio for the net is computed by using the sum of the diode-protection values of the diodes, $(0.5+1.0+1.5) * 200 + 2000 = 2600$.

Example for Diode Modes 5 and 6

Assume that the diode ratio vector is {0.0 0 1 0}; the layerMaxRatio value is 400; the gate area of an input pin is 0.6; and the following diodes are connected to a single net: diode A with a diode-protection value of 0.5, diode B with a diode-protection value of 1.0, and diode C with a diode-protection value of 1.5.

For diode modes 5 and 6, the maximum antenna ratio is computed by dividing the metal area by the sum of the gate area plus the equivalent gate area, where the equivalent gate area is computed by using the diode ratio vector.

- For diode mode 5, the equivalent gate area is computed using the maximum diode protection, which is 1.5, so the maximum antenna ratio for the net is

$$\text{metal_area} / 0.6 + ((1.5 + 0) * 1 + 0) = \text{metal_area}/2.1$$
- For diode mode 6, the equivalent gate area is computed using the total diode protection, which is $0.5+1.0+1.5=3.0$, so the maximum antenna ratio for the net is

$$\text{metal_area} / 0.6 + ((3.0 + 0) * 1 + 0) = \text{metal_area}/3.6$$

Example for Diode Modes 7 and 8

Assume that the diode ratio vector is {0.7 0.0 150 800}; the layerMaxRatio value is 400; and the following diodes are connected to a single net: diode A with a diode-protection value of 0.5, diode B with a diode-protection value of 1.0, and diode C with a diode-protection value of 1.5.

For diode modes 7 and 8, the maximum antenna ratio is computed by dividing the metal area minus the equivalent metal area by the gate area, where the equivalent metal area is computed by using the diode ratio vector.

- For diode mode 7, the equivalent metal area is computed using the maximum diode protection, which is 1.5, so the maximum antenna ratio for the net is

$$(\text{metal_area} - ((1.5 + 0) * 150 + 800)) / \text{gate_area} = (\text{metal_area} - 1025) / \text{gate_area}$$
- For diode mode 8, the equivalent metal area is computed using the total diode protection, which is $0.5+1.0+1.5=3.0$, so the maximum antenna ratio for the net is

$$(\text{metal_area} - ((3.0 + 0) * 150 + 800)) / \text{gate_area} = (\text{metal_area} - 1250) / \text{gate_area}$$

Reporting Antenna Rules

You can get a report of the metal layer antenna rules by using the `report_antenna_rules` command. For example,

```
icc_shell> report_antenna_rules -output ./antenna.rule design
```

Removing Antenna Rules

You can remove the metal layer antenna rules you set by using the `remove_antenna_rules` command. For example,

```
icc_shell> remove_antenna_rules design
```

Specifying Antenna Properties

You specify the pin and port antenna properties either in a cell library format (CLF) file or by using the `set_route_zrt_detail_options` command to set the following detail route options:

- `-default_diode_protection`
- `-default_gate_size`
- `-default_port_external_gate_size`
- `-default_port_external_antenna_area`
- `-port_antenna_mode`
- `-macro_pin_antenna_mode`

If your design contains hard macros, you must use the `signoff_calculate_hier_antenna_property` command to extract the antenna information for the hard macros.

If you are using a hierarchical flow and create a block abstraction for a block, you must use the `extract_zrt_hier_antenna_property` command to extract the antenna information from the block to use at the next level of hierarchy.

For more information about extracting the antenna properties for hard macros or block abstractions, see the *Library Data Preparation for IC Compiler User Guide*.

You can set external antenna properties for all I/O ports or for a specified I/O port by using the `define_io_gate_size`, `define_io_diode_protection`, and `define_io_antenna_area` commands. To remove the definitions set by these commands, use the `remove_io_antenna_properties` command.

You can report the antenna properties set by these commands by using the `report_io_antenna_properties` command.

Analyzing and Fixing Antenna Violations

If the Milkyway design library contains antenna rules, Zroute automatically analyzes and fixes antenna violations.

Just like other design rules, antenna rules are checked and corrected during detail routing. This concurrent antenna rule correction architecture reduces total runtime by minimizing the iterations.

By default, Zroute

- Checks antenna rules and corrects violations for all clock and signal nets

You can disable fixing of antenna violations on specific nets by using the `set_route_zrt_detail_options` command to set the `-skip_antenna_fixing_for_nets` detail route option. Note that Zroute analyzes the antenna rules and reports the antenna violations on these nets, but does not fix the violations.

- Does not check or correct antenna rules for power and ground nets

To check and correct antenna rules for power and ground nets, use the `set_route_zrt_detail_options` command to set the `-check_antenna_on_pg` detail route option to `true`.

- Starts fixing antenna violations in the second iteration, after initial routing is complete and the basic DRC violations have been fixed

You can change the iteration in which Zroute starts fixing antenna violations by using the `set_route_zrt_detail_options` command to set the `-antenna_on_iteration` detail route option.

- Performs layer hopping to fix antenna violations

Layer hopping decreases the antenna ratio by splitting a large metal polygon into several upper-level polygons. Zroute performs the following types of layer hopping:

- Breaking the antenna with a higher-level metal segment

Zroute uses this technique to fix most antenna violations. For antenna violations that happen at metal-N, inserting a small segment of metal-(N+1) close to the gate reduces the ratio between the remaining metal-N, making the ratio much lower. This approach is not suitable for fixing top-metal layer antenna violations when the output pin can provide only limited protection because there is no way for the router to break antenna violations at the topmost metal layer.

- Moving down to a lower-level metal

Zroute uses this technique to fix only topmost layer antenna violations when output pins provide only limited protection. For antenna violations that happen at metal-N,

replace part of the metal-N with metal-(N-1 or lower) to reduce the ratio. However, splitting the metal layer into many pieces might have a negative impact on RC and timing delay.

Zroute can also insert diodes to fix antenna violations. To enable the insertion of diodes to fix antenna violations, use the `set_route_zrt_detail_options` command to set the `-insert_diodes_during_routing` detail route option to `true`. To force Zroute to fix antenna violations by inserting diodes, you can disable layer hopping by using the `set_route_zrt_detail_options` command to set the `-hop_layers_to_fix_antenna` detail route option to `false`. For information about inserting diodes to fix antenna violations, see “[Inserting Diodes During Detail Routing” on page 7-22](#).

If both layer hopping and diode insertion are enabled, by default, Zroute first tries to use layer hopping to fix the antenna violation. To change the preference to diode insertion, use the `set_route_zrt_detail_options` command to set the `-antenna_fixing_preference` detail route option to `use_diodes`.

As with other design rule violations, antenna violations are reported at the end of each detail routing iteration. For example,

DRC-SUMMARY:

```
@@@@@@@ TOTAL VIOLATIONS =      506
     @@@@ Total number of instance ports with antenna violations =    1107
```

To disable the analysis and correction of antenna rules during detail routing, use the `set_route_zrt_detail_options` command to set the `-antenna` detail route option to `false`.

```
icc_shell> set_route_zrt_detail_options -antenna false
```

To check for antenna violations, use the `verify_zrt_route -antenna` command. Due to the different connectivity models, do not use the `report_antenna_ratio` command on designs routed with Zroute. You can also run the `report_design -physical` command to report information about antenna violations.

Inserting Diodes During Detail Routing

One way to protect gates from antenna effects is to provide a discharge path for the accumulated charge to leave the net. However, the discharge path should not allow current to flow during normal chip operation. Discharging can be accomplished by inserting a reverse-biased diode on the net close to the gate that is being protected.

To control diode insertion during detail routing, use the `set_route_zrt_detail_options` command to set various detail route options.

To enable diode insertion during detail routing, set the `-insert_diodes_during_routing` detail route option to `true`. To specify a preference for fixing antenna violations by using diode insertion, set the `-antenna_fixing_preference` detail route option to `use_diodes`.

To require fixing of antenna violations by using diode insertion, you must disable layer hopping by setting the `-hop_layers_to_fix_antenna` detail route option to `false`.

By default, when you enable diode insertion, Zroute can fix an antenna violation either by adding a new diode or by using an existing spare diode. Zroute determines which method to use, based on which is closest to the required location: an empty location for a new diode or an existing spare diode. You can specify a preference for using a new diode or using an existing spare diode by setting the `-diode_preference` detail route option to `new` or `spare`, respectively.

Note:

To take advantage of spare diodes for antenna violation fixing, you need to add the spare diodes either before or after standard-cell placement and before routing the areas where antenna violations might occur. For details, see [Chapter 13, “ECO Flow.”](#)

If you want Zroute to use only one of these methods, set the `-diode_insertion_mode` detail route option to `new` to force the insertion of new diodes or to `spare` to force the use of existing spare diodes. To reset the diode insertion method to the default behavior, set the `-diode_insertion_mode` detail route option to `new_and_spare`.

When inserting new diodes, Zroute selects the diodes from the reference library and inserts them into existing open spaces. You can control which diodes are used by setting the `-diode_libcell_names` detail route option. By default, Zroute reuses existing filler cell locations for diode insertion. You can prevent Zroute from reusing these locations by setting the `-reuse_filler_locations_for_diodes` detail route option to `false`.

Zroute considers voltage areas when inserting diode cells and also observes the logic hierarchy assignments for diode cells.

- If a pin has an antenna violation, the diode cells are inserted at the same level of logic hierarchy as the violating pin.
- If a top-level port has an antenna violation, by default, the diode cells are inserted at the top level. However, if the port belongs to a voltage area, you can insert the diode cells in the logic hierarchy associated with the voltage area by setting the `-use_lower_hierarchy_for_port_diodes` detail route option to `true`.

For information about setting the detail route options, see [“Setting Detail Route Options”](#) on page 6-57.

Inserting Diodes After Detail Routing

You can fix antenna violations after detail routing by explicitly specifying which violations to fix and providing constraints for fixing them. Based on the specified constraints and the setting of the `-diode_insertion_mode` detail route option, Zroute either inserts new diodes or reuses existing spare diodes to fix the specified violations.

To insert diodes after detail routing, use the `insert_zrt_diodes` command. When you use this command, you must specify the location of each antenna violation to fix by specifying the port and cell instance, the reference cell for the diode, the number of diodes to insert, the highest allowed routing layer used for connecting the diode, and the maximum distance from the specified pin that the diode can be inserted.

You use the following format to specify these values:

```
{port_name instance_name diode_reference number_of_diodes
    max_routing_layer max_routing_distance}
```

Note:

You can use this command to insert diodes for top-level ports by specifying the name of the top-level design for the cell instance.

If a diode cannot be inserted or reused within the specified distance, the tool does not insert a diode for that violation.

The `insert_zrt_diodes` command uses detail route options set with the `set_route_zrt_detail_options` command to control diode insertion. Use the `-diode_insertion_mode` detail route option to control whether Zroute inserts new diodes or reuses spare diodes. Use the `-reuse_filler_locations_for_diodes` detail route option to control whether filler cell locations can be reused. Use the `-use_lower_hierarchy_for_port_diodes` detail route option to determine the logic hierarchy in which to insert the diodes for top-level ports. For information about setting the detail route options, see “[Setting Detail Route Options](#)” on page 6-57.

For example, to insert 2 Adiode diode cells to fix an antenna violation on the A port of the CI cell instance, where the diodes must be inserted using no higher layer than Metal5 and within 2.5 microns of the port, enter the following command:

```
icc_shell> insert_zrt_diodes {{A CI Adiode 2 metal5 2.5}}
```

Removing Diodes

To remove diodes, use the `remove_diode` command or choose Finishing > Remove Diodes in the GUI. The `remove_diode` command first disconnects all the diode ports on all the specified nets; then it removes the diode cells that are fully disconnected. For example,

```
icc_shell> remove_diode -nets net_name
icc_shell> remove_diode -nets [get_nets -hierarchical net_name]
```

Inserting Redundant Vias

Redundant via insertion is an important design-for-manufacturing (DFM) feature that is supported by Zroute throughout the routing flow. In each routing stage, Zroute concurrently optimizes via count as well as wire length. The redundant via result is measured by the redundant via conversion rate, which is defined as the percentage of single vias converted into redundant vias. You should also pay attention to the number of unoptimized single vias. If a design has fewer unoptimized single vias, it is usually better for DFM. The following sections describe how to insert redundant vias:

- [Inserting Redundant Vias on Clock Nets](#)
 - [Inserting Redundant Vias on Signal Nets](#)
-

Inserting Redundant Vias on Clock Nets

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 [“Postroute Redundant Via Insertion” on page 7-29](#).

To insert redundant vias on clock nets during clock routing,

1. Specify 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 2x1 R} {V12 2x1 NR} {V12 1x2 R} {V12 1x2 NR} \
    {V23 2x1 R} {V23 2x1 NR} {V23 1x2 R} {V23 1x2 NR}}
```

Be sure to consider both DFM and routing when you select the redundant vias; otherwise, if you select the redundant vias based only on DFM considerations, you could negatively impact the routability.

In addition to using nondefault routing rules to define the redundant vias for clock nets, you can also use them to define stricter wire width and spacing rules and to define the tapering distance. For more information about nondefault routing rules, see [“Using Nondefault Routing Rules” on page 6-16](#).

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> 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
```

Zroute reserves space for the redundant vias during global routing and inserts the redundant vias during detail routing.

Inserting Redundant Vias on Signal Nets

You can perform redundant via insertion in the following ways:

- Postroute redundant via insertion
- Concurrent soft-rule-based redundant via insertion
- Near 100 percent redundant via insertion

In general, you should start with postroute redundant via insertion. If postroute redundant via insertion results in a redundant via rate of at least 80 percent, you can try to improve the redundant via rate by using concurrent soft-rule-based redundant via insertion. If postroute redundant via insertion results in a redundant via rate of at least 90 percent, you can try to improve the redundant via rate by using near 100 percent redundant via insertion.

Note:

As the redundant via rate increases, it becomes more difficult to converge on the routing design rules and you might see a reduction in signal integrity; therefore, you should use near 100 percent redundant via insertion only for those designs that truly require such a high redundant via rate. In addition, achieving very high redundant via rates might require you to modify the floorplan utilization to allow enough space for the redundant vias.

The following sections describe the default via mapping table, how to define a customized via mapping table, how to insert redundant vias by using various methods, and how to report the redundant via rate.

Viewing the Default Via Mapping Table

By default, Zroute reads the default contact codes from the technology file and generates an optimized via mapping table. In most cases you achieve better results if you use a customized mapping table rather than the default mapping table. The following section, “[Defining a Customized Via Mapping Table](#)” describes how to define a customized mapping table.

If you have not previously defined a customized mapping table for the design, you can see the default mapping table by using the `insert_zrt_redundant_vias -list_only`

command (or choose Route > Insert Redundant Vias in the GUI and select “List redundant via mappings”).

Note:

After you have created a customized mapping table by using the method described in the following section, “[Defining a Customized Via Mapping Table](#),” this command shows the customized mapping table.

[Example 7-1](#) shows an example of a default via mapping table.

Example 7-1 Default Via Mapping Table

```
icc_shell> insert_zrt_redundant_vias -list_only
...
#
# The currently defined command

define_zrt_redundant_vias \
    -from_via { VIA12A VIA12B VIA23 VIA34 VIA45 VIA56 } \
    -to_via { VIA12A VIA12A VIA23 VIA34 VIA45 VIA56 } \
    -to_via_x_size { 2 2 2 2 2 2 } \
    -to_via_y_size { 1 1 1 1 1 1 } \
    -to_via_weights { 1 1 1 1 1 1 } \

#
# The define command with all possible mappings (not recommended)

#define_zrt_redundant_vias \
# -from_via { VIA12A VIA12A VIA12B VIA12B VIA23 VIA34 VIA45 VIA56 \
#             VIA12f VIA23f VIA34f VIA45f VIA56f } \
# -to_via { VIA12A VIA12B VIA12A VIA12B VIA23 VIA34 VIA45 VIA56 \
#             VIA12f VIA23f VIA34f VIA45f VIA56f } \
# -to_via_x_size { 2 2 2 2 2 2 2 2 } \
#     2 2 2 2 2 } \
# -to_via_y_size { 1 1 1 1 1 1 1 1 } \
#     1 1 1 1 1 } \
# -to_via_weights { 1 1 1 1 1 1 1 1 } \
#     1 1 1 1 1 } \
```

Defining a Customized Via Mapping Table

To define a customized mapping table, use the `define_zrt_redundant_vias` command (or choose Route > Define Redundant Vias in the GUI). The mappings defined by the `define_zrt_redundant_vias` command are saved in the Milkyway design database. When you run the `define_zrt_redundant_vias` command, any existing mappings are overwritten; only the latest mappings are saved.

The vias listed in the `-from_via` and `-to_via` options can be via masters defined in the technology file or design-specific via masters created by the `create_via_master` command. The vias listed in the `-from_via` option must be parameter-based via masters. The vias listed in the `-to_via` option can be parameter-based via masters or H-shaped

rectangle-based via masters. For information about creating design-specific via masters, see “[Creating Design-Specific Via Masters](#)” on page 6-6.

By default, all mappings have the same priority, and Zroute selects the redundant vias to use based on routability. You can specify preferred redundant via mappings by using the `-to_via_weights` option to assign a weight value between 1 and 10 to each redundant via in the mapping table. During redundant via insertion, Zroute uses the higher weighted redundant vias first.

The syntax of the `define_zrt_redundant_vias` command is

```
define_zrt_redundant_vias
[-from_via {list_of_from_vias}]
[-from_via_x_size {list_of_number_of_contacts}]
[-from_via_y_size {list_of_number_of_contacts}]
[-from_via_array_mode off | swap | rotate | all]
[-to_via {list_of_to_vias}]
[-to_via_x_size {list_of_number_of_contacts}]
[-to_via_y_size {list_of_number_of_contacts}]
[-to_via_weights {list_of_weights}]
```

[Example 7-2](#) shows an example of using the `define_zrt_redundant_vias` command to define a customized via mapping table. Note that the weights must be integers between 1 and 10, where 1 is the lowest weight and 10 is the highest weight. Via mappings that have a higher weight are preferred over via mappings that have a lower weight.

Example 7-2 Customized Via Mapping Table

```
icc_shell> define_zrt_redundant_vias \
    -from_via  { VIA12 VIA12 VIA23 VIA23 VIA34 VIA34 VIA45 VIA56S \
                  VIA6STG VIATGAL VIA12E VIA12E VIA23E VIA23E VIA34E \
                  VIA34E VIA45E VIA56SE VIA12T VIA12T VIA23T VIA23T \
                  VIA34T VIA34T } \
    -to_via    { VIA12T VIA12 VIA23T VIA23 VIA34T VIA34 VIA45 VIA56S \
                  VIA6STG VIATGAL VIA12T VIA12 VIA23T VIA23 VIA34T \
                  VIA34 VIA45 VIA56S VIA12T VIA12 VIA23T VIA23 \
                  VIA34T VIA34 } \
    -to_via_x_size { 2 2 2 2 2 2 2 2 \
                     2 2 2 2 2 2 \
                     2 2 2 2 2 2 \
                     2 2 } \
    -to_via_y_size { 1 1 1 1 1 1 1 1 \
                     1 1 1 1 1 1 \
                     1 1 1 1 1 1 \
                     1 1 } \
    -to_via_weights { 5 1 5 1 5 1 1 1 \
                      1 1 1 1 1 1 \
                      1 1 1 1 1 1 \
                      1 1 }
```

To see the via mapping table associated with a design, use the `insert_zrt_redundant_vias -list_only` command (or choose Route > Insert Redundant Vias in the GUI and select “List redundant via mappings”). If you have defined a customized mapping table for the design, this command shows the customized mapping table; otherwise, it shows the default mapping table.

Postroute Redundant Via Insertion

To perform postroute redundant via insertion, use the `insert_zrt_redundant_vias` command (or choose Route > Insert Redundant Vias in the GUI).

This command can replace single-cut vias with multiple-cut via arrays, single-cut vias with other single-cut vias that have a different contact code, and multiple-cut via arrays with different multiple-cut via arrays. During redundant via insertion, the detail router also checks the design rules within the neighboring partition to minimize DRC violations.

By default, the `insert_zrt_redundant_vias` command inserts redundant vias on all nets. To insert redundant vias only on specific nets, use the `-nets` option to specify the nets. Using net-specific redundant via insertion allows you to further improve the optimized via rate without causing large-scale routing and timing changes.

After the vias are checked and replaced, the detail router rechecks for DRC violations and runs iterations to correct any violations. Specify the number of detail routing iterations by using the `set_route_zrt_detail_options` command to set the `-max_number_iterations` detail route option.

Note:

When you insert redundant vias in a multicorner-multimode design, Zroute uses only the current scenario.

If the percentage of redundant vias is not high enough, you can increase the effort level by using the `-effort` option to get a better redundant via rate. Increasing the effort level to high can increase the redundant via rate by about 3 to 5 percent by shifting the vias to make room for additional vias. However, because high-effort redundant via insertion moves the vias more, it can result in a less lithography-friendly pattern at the 45-nm technology node and below. In this case, you should use concurrent soft-rule-based redundant via insertion to improve the redundant via rate.

You can also try to increase the postroute redundant via rate by using the `set_route_zrt_detail_options` command to set the `-optimize_wire_via_effort_level` detail route option to `high`, which reduces the number of single vias and makes more room for redundant vias by reducing wire length.

After you perform the initial postroute redundant via insertion, use the `set_route_zrt_common_options` command to set the `-post_detail_route_redundant_via_insertion` common route option to enable

automatic insertion of redundant vias after subsequent detail routing or ECO routing. This helps to maintain the redundant via rate in your design.

You can also perform redundant via insertion during the `route_opt` flow by using the `set_route_zrt_common_options` command to set the `-post_detail_route_redundant_via_insertion` common route option before running `route_opt`.

Concurrent Soft-Rule-Based Redundant Via Insertion

Soft-rule-based redundant via insertion can improve the redundant via rate by reserving space for the redundant vias during routing. You can use concurrent soft-rule-based redundant via insertion during both initial routing and ECO routing. The actual via insertion is not done during routing; you must still perform postroute redundant via insertion by using the `insert_zrt_redundant_vias` command.

Note:

Reserving space during routing increases the routing runtime. You should use this method only when needed to improve the redundant via rate beyond that provided by postroute redundant via insertion and the postroute approach resulted in a redundant via rate of at least 80 percent.

To perform concurrent soft-rule-based redundant via insertion,

1. (Optional) Define the via mapping table as described in “[Defining a Customized Via Mapping Table](#)” on page 7-27.
2. Enable concurrent soft-rule-based redundant via insertion.

By default, concurrent redundant via insertion is disabled.

- To enable concurrent soft-rule-based redundant via insertion during initial routing, use the `set_route_zrt_common_options` command to set the `-concurrent_redundant_via_mode` common route option to `reserve_space`.

```
icc_shell> set_route_zrt_common_options \
           -concurrent_redundant_via_mode reserve_space
```

You can control the effort used to reserve space for the redundant vias during initial routing by using the `set_route_zrt_common_options` command to set the `-concurrent_redundant_via_effort_level` common route option. By default, Zroute uses low effort. The higher effort levels result in a better redundant via conversion rate at the expense of runtime. The low and medium efforts affect only global routing and track assignment, while high effort also affects detail routing, which can impact design rule convergence.

Note:

If the `placer_enable_enhanced_router` variable is `true` (the default) and you enable the `concurrent_redundant_via_mode` option before running the

`place_opt -congestion` command, the redundant vias are considered during congestion estimation.

- To enable concurrent soft-rule-based redundant via insertion during ECO routing, use the `set_route_zrt_common_options` command to set the `-eco_route_concurrent_redundant_via_mode` common route option to `reserve_space`.

```
icc_shell> set_route_zrt_common_options \
-eco_route_concurrent_redundant_via_mode reserve_space
```

You can control the effort used to reserve space for the redundant vias during ECO routing by using the `set_route_zrt_common_options` command to set the `-eco_route_concurrent_redundant_via_effort_level` common route option.

Note:

Using concurrent soft-rule-based redundant via insertion during ECO routing can impact timing and design rule convergence. In general, you should use this method only when you used near 100 percent redundant via insertion during initial routing.

3. Route the design.

During routing, Zroute reserves space for the redundant vias and fixes hard design rule violations.

4. Perform postroute redundant via insertion.

During postroute redundant via insertion, Zroute inserts the redundant vias in the reserved locations.

Near 100 Percent Redundant Via Insertion

You can achieve a redundant via rate near 100 percent by using hard-rule-based redundant via insertion. Hard-rule-based redundant via insertion can improve the redundant via rate by treating redundant vias as hard design rules during routing. You can use nearly 100 percent redundant via insertion only during initial routing; this method is not supported during ECO routing. When you use near 100 percent redundant via insertion during initial routing, you should use soft-rule-based redundant via insertion during ECO routing to preserve the redundant via rate achieved during initial routing.

Note:

This method can result in a very large runtime increase for congested designs. You should use this method only when needed to improve the redundant via rate beyond that provided by concurrent soft-rule-based redundant via insertion and the soft-rule-based approach resulted in a redundant via rate of at least 90 percent.

To perform concurrent hard-rule-based redundant via insertion,

1. (Optional) Define the via mapping table as described in [“Defining a Customized Via Mapping Table” on page 7-27](#).
2. Enable nearly 100 percent via insertion by using the `set_route_zrt_common_options` command to set the `-concurrent_redundant_via_mode` common route option to `insert_at_high_cost`. (By default, concurrent redundant via insertion is disabled.)

```
icc_shell> set_route_zrt_common_options \
-concurrent_redundant_via_mode insert_at_high_cost
```

You can control the effort used to reserve space for the redundant vias by using the `set_route_zrt_common_options` command to set the `-concurrent_redundant_via_effort_level` common route option.

Note:

If you enable the `concurrent_redundant_via_mode` option before running the `place_opt -congestion` command, the redundant vias are considered during congestion estimation.

3. Route the design.

During routing, Zroute inserts the redundant vias and fixes hard design rule violations. In general, redundant via insertion has the same priority as other hard design rules; however, if design rule checking does not converge during detail routing, Zroute automatically relaxes the redundant via constraints to improve DRC convergence.

Preserving Timing During Redundant Via Insertion

When you insert redundant vias, it changes the timing of your design. Short nets tend to slow down due to increased capacitance, whereas long nets tend to speed up due to decreased resistance. Zroute redundant via insertion has a timing-preservation mode that allows you to perform redundant via insertion without affecting the design timing by preventing insertion of redundant vias on critical nets.

To enable timing-preservation mode for redundant via insertion, define the timing preservation constraints by using the following options of the `insert_zrt_redundant_vias` command:

- `-timing_preserve_setup_slack_threshold`
- `-timing_preserve_hold_slack_threshold`
- `-timing_preserve_nets`

You should timing-preservation mode only at the end of the flow, after using the normal redundant via insertion flows, which converge both the redundant via rate and the timing QoR. Timing-preservation mode can slightly increase the redundant via rate while maintaining timing. However, if you use timing-preservation mode earlier in the flow, before timing is met, it might severely reduce the redundant via rate due to critical nets.

Maximizing the Redundant Via Rate

To maximize the redundant-via insertion rate, use the following flow to insert redundant vias:

1. (Optional) Define the via mapping (`define_zrt_redundant_vias`).
2. Route the design (`route_opt -initial_route_only`).
3. Perform postroute redundant via insertion (`insert_zrt_redundant_vias`).

For details about the postroute redundant via insertion flow, including how to increase the redundant via rate when using this flow, see “[Postroute Redundant Via Insertion](#)” on page 7-29.

If postroute redundant via insertion achieves an 80 percent redundant via rate and you need a higher rate, you can try soft-rule-based concurrent redundant via insertion. For details about this flow, see “[Concurrent Soft-Rule-Based Redundant Via Insertion](#)” on page 7-30.

If postroute redundant via insertion achieves a 90 percent redundant via rate and you need a higher rate, you can try near 100 percent redundant via insertion. For details about this flow, see “[Near 100 Percent Redundant Via Insertion](#)” on page 7-31.

4. Enable concurrent redundant via insertion to maintain the redundant via rate during additional routing operations (`set_route_zrt_common_options -post_detail_route_redundant_via_insertion medium`).
If you used near 100 percent redundant via insertion during initial routing, you should also enable soft-rule-based concurrent redundant via insertion during ECO routing (`set_route_zrt_common_options -eco_route_concurrent_redundant_via_mode reserve_space`).
5. Perform postroute optimization (`route_opt -skip_initial_route`).
6. Perform DFM tasks.
7. Perform incremental postroute optimization (`route_opt -incremental -size_only`)
8. Insert redundant vias using timing-preservation mode.

Reporting Redundant Via Rates

After redundant via insertion, whether concurrent or postroute, Zroute generates a redundant via report that provides the following information:

- The via conversion rate for nondefault vias

The via conversion rate for nondefault vias is listed at the top of the report as the total optimized via conversion rate.

- The optimized via conversion rate for each layer

The optimized via conversion rate includes both double vias and DFM-friendly bar vias, which have a single cut but a larger metal enclosure.

Note:

The optimized via conversion rate is not useful if you are using bar vias.

- The distribution of optimized vias by weight for each layer

To determine the via conversion rate for conversions above a certain weight, you must add the reported conversion rates for those weights. For example, in [Example 7-3](#), the via conversion rate for weight 5 and above for layer V03 is $10.75+64.50=75.25\%$.

For information about assigning weights to redundant via mappings, see “[Defining a Customized Via Mapping Table](#)” on page [7-27](#).

Note:

The conversion rate for unweighted vias is reported as “Un-optimized.”

- The total double via conversion rate for the design

[Example 7-3](#) shows an example of the redundant via report.

Example 7-3 Redundant Via Report

```
Total optimized via conversion rate = 96.94% (1401030 / 1445268 vias)
Layer V01      = 41.89% (490617 / 1171301 vias)
  Weight 10    = 9.64% (112869 vias)
  Weight 5     = 32.25% (377689 vias)
  Weight 1     = 0.01% (59 vias)
  Un-optimized = 58.11% (680684 vias)
Layer V02      = 76.20% (1567822 / 2057614 vias)
  Weight 10    = 43.51% (895270 vias)
  Weight 5     = 28.62% (588805 vias)
  Weight 1     = 4.07% (83747 vias)
  Un-optimized = 23.80% (489792 vias)
Layer V03      = 81.87% (687115 / 839297 vias)
  Weight 10    = 64.50% (541369 vias)
  Weight 5     = 10.75% (90224 vias)
  Weight 1     = 6.62% (55522 vias)
  Un-optimized = 18.13% (152182 vias)
Layer V04      = 81.60% (226833 / 277977 vias)
  Weight 10    = 81.45% (226418 vias)
  Weight 1     = 0.15% (415 vias)
  Un-optimized = 18.40% (51144 vias)
...
Layer V09      = 85.47% (1329 / 1555 vias)
  Weight 10    = 85.47% (1329 vias)
  Un-optimized = 14.53% (226 vias)

Total double via conversion rate = 46.69% (2158006 / 4622189 vias)
```

You can also use the `report_design -physical` command to report the double via rate. This command reports the double via rate per layer, but does not provide any weighting information.

Optimizing Wire Length and Via Count

During detail routing, Zroute optimizes wire length and via count in the areas where DRC violations occur; however, it does not optimize the layout in areas where no DRC violations occur.

To improve the manufacturing yield, you can use the `optimize_zrt_wire_via` command to perform standalone optimization of wire length and via count after performing detail routing and redundant via insertion.

By default, Zroute selects the nets to reroute based on the overall cost. For each selected net, Zroute determines whether to reroute all the shapes in the net or just a portion of them. If you want to select the nets to optimize, use the `-nets` option to specify the nets. When you specify the nets to optimize, you can also use the `-reroute_all_shapes_in_nets` option to control whether Zroute must reroute all the associated net shapes.

By default, Zroute performs a maximum of 40 detail routing iterations to fix DRC violations that exist after the optimization. You can use the `-max_detail_route_iterations` option to control the maximum number of detail routing iterations.

Reducing Critical Areas

A critical area is a region of the design where, if the center of a random particle defect falls there, the defect causes circuit failure, thereby reducing yield. A conductive defect causes a short fault, and a nonconductive defect causes an open fault.

The following sections describe how to

- Report critical areas
- Display critical area maps
- Reduce critical area short faults by performing wire spreading
- Reduce critical area open faults by performing wire widening

Reporting Critical Areas

After routing is complete, you can report layout-critical areas that are susceptible to random particle defects, which cause shorts and opens during the fabrication process.

The results from the critical area analysis report are output to an `output_heatmap` text file. You can see the critical area results graphically by displaying critical area heat maps.

To report critical areas, use the `report_critical_area` command (or choose Finishing > Report Critical Area Map in the GUI).

[Table 7-5](#) describes the `report_critical_area` options.

Table 7-5 report_critical_area Command Options

Command option	Description
<code>-fault_type short open</code> ("Fault type" options in the GUI)	Specifies the type of fault to check for. The default is <code>short</code> .
<code>-particle_distr_func_file</code> <code>file_name</code> ("Particle distribution function file" box in the GUI)	The name of the input file that contains the particle distribution function to be used during the calculation. By default, Zroute uses an internal particle probability function.

Table 7-5 report_critical_area Command Options (Continued)

Command option	Description
-suppress_zeros_in_report ("Suppress zeros in report" check box in the GUI)	Prevents reporting of zero results. By default, all results are reported.
-multiparticle_report_format true false ("Multi particle report format" check box in the GUI)	Outputs a report containing the critical area analysis result of each individual particle size. By default, Zroute outputs normalized results.
-tsmc_encr_particle_distr_file ("TSMC encrypted format" check box in the GUI)	Specifies that the particle distribution function file is in TSMC encrypted format.
-layer_alias_DSD_format {X Y Z T R} ("Metal scheme directives" boxes in the GUI)	Specifies the layer alias used for defect size distribution (DSD) of each layer.
-input_layers layers ("Input layer" list in the GUI)	Specifies the metal layers for which the critical area is calculated. By default, the critical area is calculated for all metal layers.

Often the particle probability function is considered sensitive data. When security for a sensitive particle distribution file from a foundry is of concern, use the `process_particle_probability_file` command, which provides a way to encrypt and decrypt the particle probability function. Critical area analysis can work with such an encrypted particle probability function.

A secret key is used to encrypt the particle probability function. The encrypted file cannot be decrypted without the key. To encrypt and decrypt a particle probability function file, use the `process_particle_probability_file` command.

Critical area analysis takes the encrypted file as its input and processes it without the key. The output is the heat map based on the encrypted particle probability function; it is in text format. The text format of the particle probability function is still accepted as input to critical area analysis. If an encrypted file is given as input, the file is internally decrypted and used.

Critical area analysis takes the encrypted file as its input and processes it without the key. The output is the heat map based on the encrypted particle probability function; it is in text format. The text format of the particle probability function is still accepted as input to critical area analysis. If an encrypted file is given as input, the file is internally decrypted and used.

Displaying Critical Area Maps

Critical area maps provide an indication of places where a chip might fail due to particle defects, causing shorts and opens.

To display a critical area map for the current design,

1. Specify the type of defect to display by choosing one of the following:
 - Finishing > Short Critical Area Map
 - Finishing > Open Critical Area MapThe Map Mode panel appears.
2. Select a metal layer for which critical area shorts or opens are to be displayed.
3. Modify the critical heat ranges by changing or removing the maximum or minimum threshold values.
4. Adjust other display parameters. You can make text visible in the map.
5. Click Apply.

To update the critical area map data after you perform wire spreading, click Reload. This action opens the (Re)Calculate Short Critical Area Map Data dialog box, from which you can run the `report_critical_area -fault_type short` command.

To update the critical area map data after you perform wire widening, click Reload. This action opens the (Re)Calculate Open Critical Area Map Data dialog box, from which you can run the `report_critical_area -fault_type open` command.

For more information about using critical area maps, see the “Examining Critical Area Maps” topic in IC Compiler Help.

Performing Wire Spreading

After you have performed detail routing and redundant via insertion, you can perform wire spreading to increase the average spacing between wires, which reduces the critical area short faults and therefore improves yield.

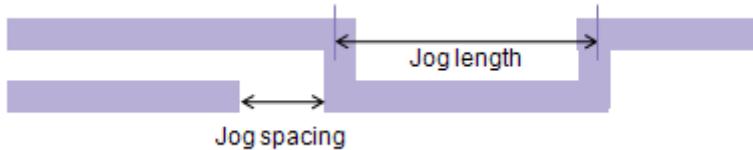
To perform wire spreading, use the `spread_zrt_wires` command (or choose Finishing > Route Spread Wires in the GUI).

By default, the `spread_zrt_wires` command spreads the signal wires on the same layer by half a pitch in the preferred direction. You can change the spread distance by using the `-pitch` option. By default, the minimum jog length is twice the layer pitch and the minimum jog spacing is the minimum spacing for the layer plus one half the pitch for the layer. You can modify the minimum jog length by using the `-min_jog_length` option. You specify the

minimum jog length as the ratio of jog length to layer pitch. You can modify the minimum jog spacing by using the `-min_jog_spacing_by_layer_name` option. You specify the minimum jog spacing in microns for each layer.

[Figure 7-5](#) shows how the jog length and jog spacing values are used in wire spreading.

Figure 7-5 Wire Spreading Results



In the following example, the minimum jog length is set to three times the layer pitch, the minimum jog spacing for metal1 is set to 0.07 microns, and the minimum jog spacing for metal2 is 0.08 microns. All other metal layers use the default minimum jog spacing.

```
icc_shell> spread_zrt_wires -min_jog_length 3 \
    -min_jog_spacing_by_layer_name {{metal1 0.07} {metal2 0.08}}
```

After spreading, the `spread_zrt_wires` command performs detail routing iterations to fix any DRC violations caused as a result of spreading.

When you change the layout, it can change the timing of your design. Zroute wire spreading has a timing-preservation mode that allows you to perform wire spreading without affecting the design timing.

To enable timing-preservation mode for wire spreading, define the timing preservation constraints by using the following options of the `spread_zrt_wires` command:

- `-timing_preserve_setup_slack_threshold threshold`
- `-timing_preserve_hold_slack_threshold threshold`
- `-timing_preserve_nets nets`

The threshold values are floating-point numbers in library units. Wire spreading is performed only on nets with slack greater than or equal to the specified values or those specified in the `-timing_preserve_nets` option, as well as adjacent nets on the same layer within two routing pitches.

Performing Wire Widening

After you have performed detail routing, redundant via insertion, and wire spreading, you can perform wire widening to increase the average width of the wires, which reduces the critical area open faults and therefore improves yield.

To perform wire widening, use the `widen_zrt_wires` command (or choose Finishing > Route Widen Wires in the GUI).

When you perform wire widening, the spacing between neighboring wires is decreased, which can reduce the improvement in critical area shorts gained from wire spreading. You can control the tradeoff between wire spreading and wire widening by using the `-spreading_widening_relative_weight` option. By default, wire spreading and wire widening are given equal priority. To weight the priority toward wire widening and reduced critical area open faults, set this option to a value between 0.0 and 0.5. To weight the priority toward wire spreading and reduced critical area short faults, set this option to a value between 0.5 and 1.0.

By default, the `widen_zrt_wires` command widens all wires in the design to 1.5 times their original width. For more flexibility, you can use the `-widen_widths_by_layer_name` option to define up to five possible wire widths to use for each layer. For example, to define possible wire widths of 0.07 and 0.06 microns for metal1; wire widths of 0.08 and 0.07 microns for metal2; and 1.5 times the existing wire width for all other layers, enter the following command:

```
icc_shell> widen_zrt_wires \
    -widen_widths_by_layer_name {{metal1 0.07 0.06} {metal2 0.08 0.07}}
```

After spreading, the `widen_zrt_wires` command performs detail routing iterations to fix any DRC violations caused as a result of widening. Note that the widened wires do not trigger fat wire spacing rules.

When you widen the wires, it changes the timing of your design. Zroute wire widening has a timing-preservation mode that allows you to perform wire widening without affecting the design timing.

To enable timing-preservation mode for wire widening, define the timing preservation constraints by using the following options of the `widen_zrt_wires` command:

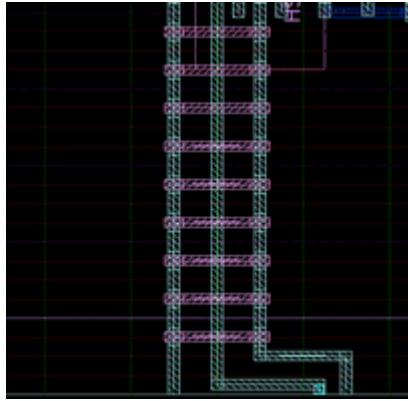
- `-timing_preserve_setup_slack_threshold threshold`
- `-timing_preserve_hold_slack_threshold threshold`
- `-timing_preserve_nets nets`

The threshold values are floating-point numbers in library units. Wire widening is not performed on nets with slack less than the specified values or those specified in the `-timing_preserve_nets` option.

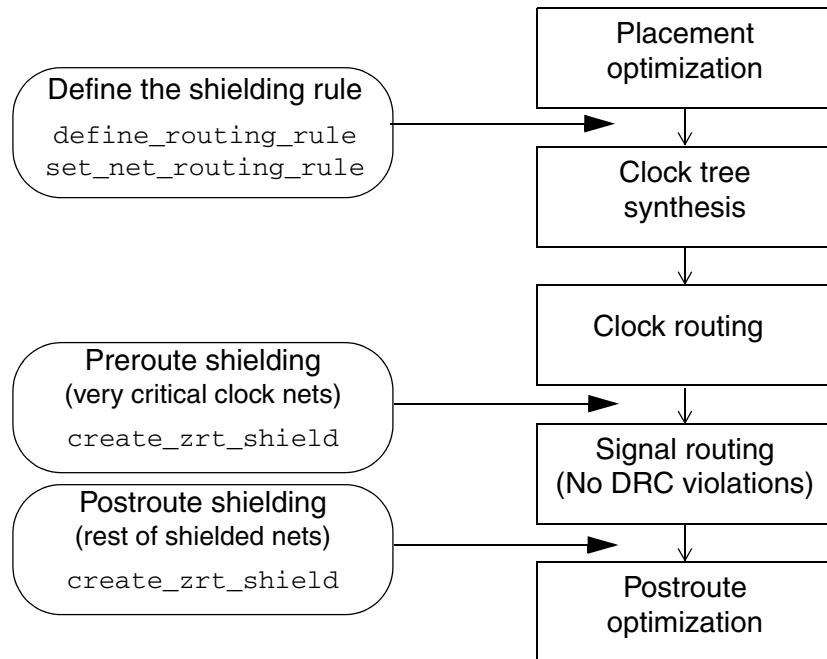
Shielding Nets

The router shields routed nets by generating shielding wires that are based on the shielding widths and spacing defined in the shielding rules. In addition to shielding nets on the same layer, you also have the option to shield one layer above or one layer below or the layer above and the layer below. Shielding above or below the layer is called *coaxial shielding*. [Figure 7-6](#) shows an example of coaxial shielding. Coaxial shielding provides even better signal isolation than same-layer shielding, but it uses more routing resources.

Figure 7-6 Coaxial Shielding



You can perform shielding either before or after signal routing. Shielding before signal routing, which is referred to as preroute shielding, provides better shielding coverage but can result in congestion issues during signal routing. Preroute shielding is typically used to shield critical clock nets. Shielding after signal routing, which is referred to as postroute shielding, has a very minimal impact on routability, but provides less protection to the shielded nets. [Figure 7-7](#) shows the Zroute shielding flow, which is described in the sections that follow.

Figure 7-7 Zroute Shielding Flow

Defining the Shielding Rules

Before you perform shielding, you must

1. Define the shielding rules.

To define shielding rules, use the `-shield_widths` and `-shield_spacings` options of the `define_routing_rule` command. For example, to specify a shielding rule that uses spacing of 0.1 microns and width of 0.1 microns for metal1 through metal5 and spacing of 0.3 microns and width of 0.3 microns for metal6, enter the following command:

```
icc_shell> define_routing_rule shield_rule \
    -shield_widths {M1 0.1 M2 0.1 M3 0.1 M4 0.1 M5 0.1 M6 0.3} \
    -shield_spacings {M1 0.1 M2 0.1 M3 0.1 M4 0.1 M5 0.1 M6 0.3}
```

For more information about the `define_routing_rule` command, see “[Defining Nondefault Routing Rules](#)” on page 6-16.

2. Assign shielding rules to the nets to be shielded.

To avoid congestion issues and achieve the best balance of DRC convergence and timing closure, you should apply shielding rules only to high-frequency or critical clock nets and apply double-spacing rules to the lower-frequency clock nets.

Use the `set_clock_tree_options` command to assign shielding rules to clock nets. For signal nets, use the `set_net_routing_rule` command to assign the shielding rules.

Note:

The `set_clock_tree_options` command assigns shielding rules to clock nets only before clock tree synthesis. After clock tree synthesis, you must use the `set_net_routing_rule` command to assign shielding rules to clock nets.

For more information about these commands, see “[Applying Nondefault Routing Rules](#)” on page 6-24.

Performing Preroute Shielding

To provide the most protection for critical clock nets, perform shielding on those nets after clock tree routing but before signal net routing.

To add shielding to the routed clock nets based on the assigned shielding rules, use the `create_zrt_shield` command (or choose Route > Create Shield in the GUI).

By default, the `create_zrt_shield` command

- Performs shielding on all nets with predefined shielding rules, except those marked as frozen

To explicitly specify the nets on which to perform shielding, use the `-nets` option.

- Ties the shielding wires to the ground net

If the design contains multiple ground nets or you want to tie the shielding wires to the power net, use the `-with_ground` option to specify the power or ground net to which to tie the shielding wires.

If the shielding wires can be tied to multiple power or ground nets, specify the nets by using the `set_route_zrt_common_options` command to set the `-shielding_nets` common route option before running the `create_zrt_shield` command.

Note:

If you specify both the `create_zrt_shield -with_ground` option and the `set_route_zrt_common_options -shielding_nets` option, the tool issues a warning message and uses the `set_route_zrt_common_options -shielding_nets` setting.

- Performs same-layer shielding

To perform coaxial shielding, use the `-coaxial_above` and `-coaxial_below` options. By default, the `create_zrt_shield` command leaves one routing track open between each used track. For coaxial shielding above the shielded net segment layer (`-coaxial_above true`), you control the number of open tracks between used tracks by using the `-coaxial_above_skip_tracks` option. For coaxial shielding below the shielded net segment layer (`-coaxial_below true`), you control the number of open tracks between used tracks by using the `-coaxial_below_skip_tracks` option. For either of these options, you can specify an integer between zero and seven.

For finer control of the routing resources used for shielding, you can specify the number of tracks to skip on a per-layer basis by using the `-coaxial_skip_tracks_on_layers` option. To disable shielding on a specific layer, set the value to -1; otherwise, specify an integer between 0 and 7 to specify the number of tracks to skip.

Note:

You cannot mix these two methods of specifying coaxial shielding; you must use either the options to control shielding above and below the net segments (`-coaxial_above`, `-coaxial_above_skip_tracks`, `-coaxial_below`, and `-coaxial_below_skip_tracks` options) or the layer-based syntax (`-coaxial_skip_tracks_on_layers` option).

If the generated coaxial shielding wires violate minimum area or minimum length rules, Zroute automatically patches the wires to satisfy these design rules.

- Connects the shielding wires to the standard-cell power or ground pins and the standard-cell rails

To prevent connections to the standard-cell power and ground pins, set the `-ignore_shielding_net_pins` option to `true`. To prevent connections to the standard-cell rails, set the `-ignore_shielding_net_rails` option to `true`.
- Creates the shielding wires such that they surround the shielded routing shape

To trim the shielding wires so that they align with the shielded routing shape ends, set the `-align_to_shape_end` option to `true`. To force Zroute to create shielding wires only in the preferred direction, set the `-preferred_direction_only` option to `true`. Note that the `-preferred_direction_only` option does not honor route guides to change the preferred routing direction. When you set either of these options to `true`, extra effort is required to connect the shielding wires to the power and ground network and any shielding wires that are not connected to the power and ground network are deleted.

For example, to perform coaxial shielding below the shielded net segment layer on the clock nets, prevent signal routing below the shielded net segment layer, and tie the shielding wires to VSS, use the following command:

```
icc_shell> create_zrt_shield -nets $clock_nets \
  -coaxial_below true -coaxial_below_skip_tracks 0 \
  -with_ground VSS
```

When you run the `create_zrt_shield` command, it reports both the net-based and length-based average shield ratios, as shown in the following example:

```
Shielded 82% side-wall of (reset)
Shielded 96% side-wall of (clk)
Shielded 2 nets with average ratio as follows.
 1) 89.00%          (total shield ratio/number of shielded nets)
 2) 87.92%          (total shield length/total shielded net length)
```

Note:

In some cases there is a slight difference in the shield ratios reported by the `create_zrt_shield` and `report_zrt_shield` commands. This is due to graph connectivity differences between the two commands. When the reported values differ, use the values reported by the `report_zrt_shield` command. The difference in reported values is typically less than three percent, but can be up to five percent.

If you use the `-preferred_direction_only true` option when running the `create_zrt_shield` command, but Zroute must use some nonpreferred direction wires for shielding, the shielding ratio report specifies the percentage of nonpreferred direction wires.

By default, the power and ground structure is not included in the shielding ratio calculation. To include the power and ground structure within a threshold distance in the shielding ratio calculation, use the `set_route_zrt_common_options` command to set the `-pg_shield_distance_threshold` common route option, which specifies the distance threshold in microns.

```
icc_shell> set_route_zrt_common_options \
    -pg_shield_distance_threshold distance
```

Note that this option affects only the shielding ratio calculation and does not change the routing behavior.

Soft Shielding Rules During Signal Routing

Zroute considers shielding rules as soft rules during signal routing. If a net has a shielding rule and is not shielded before signal routing, by default, Zroute reserves shielding space during the whole routing process: global routing, track assignment, and detail routing. At the end of detail routing, it reports the shielding space violations and the locations where shielding wires cannot be established. The following log file example shows shielding soft spacing violations, which are highlighted in bold text:

```
DRC-SUMMARY:
@@@@@@@ TOTAL VIOLATIONS = 13
Diff net var rule spacing : 2
Same net spacing : 2
Less than minimum area : 4
Short : 1
Soft spacing (shielding) : 2
```

Signal routing only reserves space for the shielding; it does not actually insert it. You must run `create_zrt_shield` after signal routing to physically place the shielding wires. The reported soft rule violations help you to understand the shielding rate. Note that the router reserves space only for same-layer shielding and not for coaxial shielding; therefore, postroute coaxial shielding can produce a very low shielding rate.

If you want to use power and ground nets for shielding, and do not want Zroute to reserve space for the shielding when power and grounds nets are available, set the `allow_pg_as_shield` common route option to `true` before running signal routing.

Performing Postroute Shielding

To perform postroute shielding, you use the same command, `create_zrt_shield`, that is used for preroute shielding.

If you want to use the default spacings and widths from the technology file for postroute shielding, you do not need to define and assign nondefault routing rules. If you specify the nets to be shielded by using the `-nets` option, `create_zrt_shield` shields these nets with the default spacing and widths.

Postroute shielding should introduce few to no DRC violations. If DRC violations are created during shielding, `create_zrt_shield` triggers five detail routing iterations to fix them. If this does not fix the DRC violations, you can fix the remaining violations by running incremental detail routing with the `route_zrt_detail -incremental` command. It is possible that postroute shielding might break some tie-off connections during the shield trimming process. In this case, use `route_zrt_eco` instead of `route_zrt_detail` to rebuild the tie-off connections and to fix the DRC violations.

Shielding Example

[Example 7-4](#) provides an example of shielding clock nets using preroute shielding and shielding critical nets using postroute shielding.

Example 7-4 Shielding Flow Example

```
# Define shielding rule
define_routing_rule shield_rule \
    -shield_widths {M1 0.1 M2 0.1 M3 0.1 M4 0.1 M5 0.1 M6 0.3} \
    -shield_spacings {M1 0.1 M2 0.1 M3 0.1 M4 0.1 M5 0.1 M6 0.3}

# Assign shielding rule to clock nets
set_clock_tree_options -clock_trees CLK \
    -routing_rule shield_rule

# Perform clock tree synthesis
set_delay_calculation_options -routed_clock arnoldi
clock_opt -no_clock_route

# Route the clock nets, reusing the global routing result from clock_opt
route_zrt_group -all_clock_nets -reuse_existing_global_route true

# Perform preroute shielding for the clock nets
create_zrt_shield -nets $clock_nets \
    -with_ground VSS
```

```

# Assign shielding rule to critical nets
set_net_routing_rule -rule shield_rule $critical_nets

# Route signal nets using shielding soft rules
route_zrt_auto

# Perform postroute shielding for critical nets
create_zrt_shield -nets $critical_nets \
    -with_ground VSS

```

Performing Incremental Shielding

By default, Zroute does not perform incremental shielding on nets that were modified after they were shielded. However, you can enable this capability for nets that were initially shielded with the `create_zrt_shield` command by using the `set_route_zrt_common_options` command to change the `-reshield_modified_nets` common route option from its default of `off`.

When you enable incremental shielding, Zroute performs incremental shielding during detail routing by removing the existing shielding from the modified nets and optionally reshielding these nets based on the new topology.

- To remove the existing shielding only, set the `reshield_modified_nets` common route option to `unshield`.
- To remove the existing shielding and reshield the modified nets, set the `reshield_modified_nets` common route option to `reshield`.

Note:

Zroute automatically detects the nets modified within the IC Compiler tool; however, nets modified externally and input by reading a DEF file are not supported by incremental shielding.

Reporting Shielding Information

After you run the `create_zrt_shield` command, you can report the shielding statistics by running the `report_zrt_shield` command.

Note:

In some cases there is a slight difference in the shield ratios reported by the `create_zrt_shield` and `report_zrt_shield` commands. This is due to graph connectivity differences between the two commands. When the reported values differ, use the values reported by the `report_zrt_shield` command. The difference in reported values is typically less than three percent, but can be up to five percent.

By default, the power and ground structure is not included in the shielding ratio calculation. To include the power and ground structure within a threshold distance in the shielding ratio

calculation, use the `set_route_zrt_common_options` command to set the `-pg_shield_distance_threshold` common route option, which specifies the distance threshold in microns.

```
icc_shell> set_route_zrt_common_options \
-pg_shield_distance_threshold distance
```

The default report generated by the `report_zrt_shield` command provides overall statistics, as shown in [Example 7-5](#).

Example 7-5 Default Shielding Report

```
icc_shell> report_zrt_shield
...
Shielded 82% side-wall of (reset)
Shielded 96% side-wall of (clk)
Shielded 2 nets with average ratio as follows.
  1) 89.00%          (total shield ratio/number of shielded nets)
  2) 87.92%          (total shield length/total shielded net length)
```

You can output the statistics for each layer by using the `-per_layer true` option with the `report_zrt_shield` command, as shown in [Example 7-6](#).

Example 7-6 Layer-Based Shielding Report

```
icc_shell> report_zrt_shield -per_layer true
...
Shielded 82% side-wall of (reset)
  Layer: M1  ratio: 0%
  Layer: M2  ratio: 40%
  Layer: M3  ratio: 85%
  Layer: M4  ratio: 80%
Shielded 96% side-wall of (clk)
  Layer: M3  ratio: 0%
  Layer: M4  ratio: 96%
  Layer: M5  ratio: 100%
Shielded 2 nets with average ratio as follows.
  1) 89.00%          (total shield ratio/number of shielded nets)
  2) 87.92%          (total shield length/total shielded net length)
```

Inserting Filler Cells

Filler cells fill gaps in the design to ensure that all power nets are connected and the spacing requirements are met.

- Before routing, you can
 - Insert standard-cell fillers
 - Insert end cap cells
- After routing, you can
 - Insert well fillers
 - Insert pad fillers

The following sections describe how to insert these cells.

Inserting Standard-Cell Fillers

You can fill empty spaces in the standard-cell rows with instances of reference filler cells to make sure all power nets are connected. One method of improving the stability of the power supply is to add decoupling capacitors as filler cells.

You insert filler cells by using the `insert_stdcell_filler` command (or by choosing Finishing > Insert Standard Cell Filler in the GUI). Zroute supports filler cells with and without metal and supports both single-height and multiheight filler cells. By default, the tool inserts standard-cell fillers in the entire design. Use the `-voltage_area` option to restrict insertion to specific voltage areas. Use the `-plan_group` option to restrict insertion to specific plan groups.

To insert filler cells using Zroute,

1. (Optional) Define the standard-cell filler rules by using the `set_left_right_filler_rule` command to define the left and right filler rules. These rules specify the filler cell to insert immediately to the left and right respectively of specific standard cells.
2. Insert filler cells with metal.

When you insert filler cells with metal, you must use the `-cell_with_metal` option to specify the reference filler cells. You connect the inserted filler cells to the power and ground (PG) networks based on the design.

If the design has a single PG network, use the `-connect_to_power` and `-connect_to_ground` options to connect the inserted filler cell to power and ground nets.

If the design has multiple PG networks, choose one of the following methods:

- Let the `insert_stdcell_filler` command automatically derive PG connections based on the IEEE 1801 Unified Power Format (UPF) information.
Do not use the `-connect_to_power`, `-connect_to_ground`, and `-pin_net` options. Note that this method only applies to designs that contain power domains in a UPF flow.
- Use the `-connect_to_power`, `-connect_to_ground`, and `-voltage_area` options or the `-pin_net` and `-voltage_area` options to connect power domains individually.

When filler cells are inserted by using the `-cell_with_metal` option, Zroute checks for DRC violations automatically:

- If your design has more than one PG network and you do not specify the PG connections, Zroute treats the unconnected pins on the filler cells as violations and removes the filler cells when checking for shorts.
- If a filler cell with metal causes DRC violations, Zroute removes it.

3. Insert filler cells without metal.

When you insert filler cells without metal, use the same methods described for inserting filler cells with metal. The only differences are

- You can make PG connections after inserting filler cells by using the `derive_pg_connection` command.
- Zroute does not check for DRC violations when you insert filler cells without metal.

To remove filler cells without metal that cause DRC violations, use the `remove_zrt_filler_withViolation` command. Because removing the filler cells can expose new violations, you sometimes need to run this command multiple times to remove all violating filler cells.

To check for DRC violations on filler cells before removing them, run the `remove_zrt_filler_withViolation` command with the `-check_only` option. This command writes a report to a file named `design_fillers_withViolation.rpt` in the current working directory. The report lists the filler cells with violations and reports the first violation for each filler cell. Note that filler cells with fixed placement are not checked by this command.

For example, to insert filler cells and connect them to the VDD power net and VSS ground net, enter the following commands:

```
icc_shell> insert_stdcell_filler -cell_with_metal $FILLER_CELL_METAL \
    -connect_to_power VDD -connect_to_ground VSS
icc_shell> insert_stdcell_filler -cell_without_metal $FILLER_CELL \
    -connect_to_power VDD -connect_to_ground VSS
```

Connecting Multiple Power and Ground Pins

You can use the `-pin_net` option of the `insert_stdcell_filler` command to make power and ground (PG) connections for decoupling capacitors with multiple power pins or for multivoltage designs. The option lets you connect PG pins to PG nets in a specific voltage area.

Note:

You cannot use the `-pin_net` option with the `-connect_to_power` or `-connect_to_ground` option.

For example, the pd1 power domain belongs to the va1 voltage area, and the pd2 power domain belongs to the va2 voltage area in the design. The following command connects the pin1 power pin to the net1 power net in the va1 voltage area and the pin2 power pin to the net2 power net in the va2 voltage area.

```
icc_shell> insert_stdcell_filler \
    -pin_net {{pin1 net1 va1}{pin2 net2 va2}}
```

Any PG pins that are not specified by the `-pin_net` option are connected to the primary PG nets based on the UPF information.

For example, if a FILLX filler has vdd and vdda power pins and one ground pin, you only need to specify the connection of the extra power pin, as shown in the following command:

```
icc_shell> insert_stdcell_filler -cell_with_metal {FILLX} \
    -pin_net {{vdda net2 va1}}
```

As a result, the tool connects the vdda power pin to the net2 power net in the va1 voltage area, the vdd power pin to the primary power net, and the ground pin to the primary ground net.

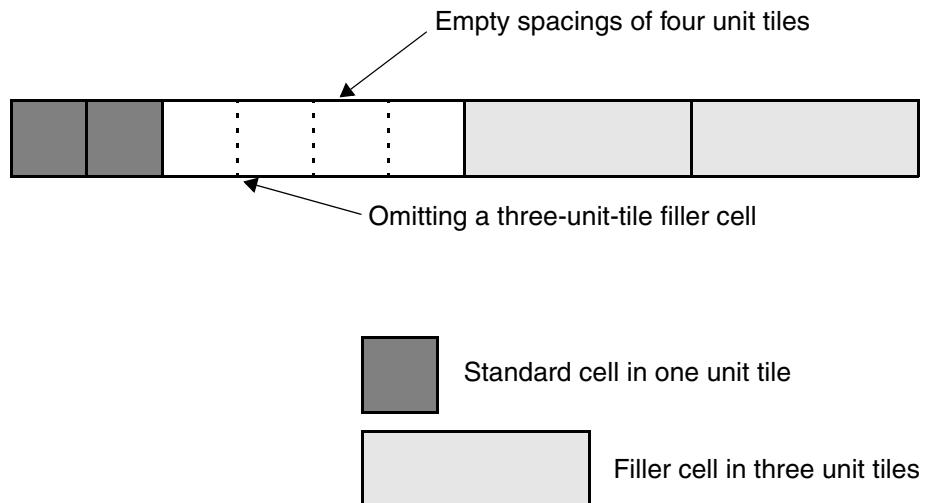
Avoiding One-Unit-Tile Violations for Overlapping Rows

When you specify the `-no_1x` option of the `insert_stdcell_filler` command, the command checks the gaps against the smallest unit tile to avoid one-unit-tile violations on overlapping rows. This behavior is aligned with the normal standard-cell legalization.

The following example sets the `-no_1x` option to prevent the command from inserting filler cells at the locations where one-unit-tile gaps might occur, as shown in [Figure 7-8](#).

```
icc_shell> insert_stdcell_filler -no_1x \
    -cell_without_metal {Fill_3 Fill_2}
```

Figure 7-8 Overlapping Row With Various Unit Tiles



When you set the `-no_1x` option, you must also use the `-cell_without_metal` option to specify the two-unit-tile and three-unit-tile reference filler cells but not the one-unit-tile reference filler cells. If you specify the one-unit-tile reference filler cell, the `-no_1x` option has no effect and the tool issues a warning message. If you do not specify the two-unit-tile and three-unit-tile reference filler cells, the tool issues an error message.

Defining the Filler Rules

The left and right filler rules specify the filler cell to insert to the immediate left and immediate right respectively of specific standard cells. You use the `set_left_right_filler_rule` command to define these rules.

If there is only a single site between two standard cells, the rule used to fill that site depends on whether the references for the standard cells are the same or different. If the references for the two standard cells are the same, the tool uses the rules for the cell on the left, and the rules for the cell on the right are ignored. If the references are different, the tool uses the rule that was defined first.

To report the right and left filler rules defined for your design, run the `report_left_right_filler_rule` command. The report shows the rules and the order in which they were defined.

By default, the left and right filler cells have a north (N) orientation or flipped-south (FS) orientation when the rows are flipped. You can choose to have the left and right filler cells follow the orientation of the standard cell by using the `set_left_right_filler_rule -follow_stdcell_orientation` option.

Reporting Filler Cells

To report the type of filler cells and their locations, use the `report_filler_placement` command. The syntax is

```
report_filler_placement -lib_cell lib_cell_list [-abut]
```

Use the `-lib_cells` option to specify the type of filler cells that you want to report in your design. To report only the adjacent filler cells that form a consecutive pair in a cell row, use the `-abut` option. For example,

```
icc_shell> report_filler_placement -lib_cell FILL1BWP -abut
```

The example reports only the consecutive filler cells named FILL1BWP and their locations in the design.

Removing Filler Cells

To remove standard-cell fillers, use the `remove_stdcell_filler -stdcell` command (or choose Finishing > Remove Fillers and select “Standard Cell” in the “Filler type” area in the GUI). By default, the filler cells are removed for the whole chip. You can optionally specify a bounding box from which to remove filler cells.

When filler cells are added after signal routing, you can remove all the filler cells that have routing design rule violations. To remove standard-cell fillers with violations, use the `remove_filler_withViolation` command (or choose Finishing > Remove Fillers With Violation in the GUI). You can restrict the removal to specific instances by using the `-name` option.

Inserting End-Cap Cells

After placing standard cells and before routing, you can add end-cap cells at both ends of a cell row. Typically, an end-cap cell is a nonlogic cell that serves a certain purpose for the row such as providing a decoupling capacitor for the power rail. Because the tool accepts any standard cell as an end-cap cell, you should specify a suitable end-cap cell.

To insert end-cap cells, use the `add_end_cap` command or choose Finishing > Insert End Cap in the GUI. You must specify the library cell to use for the end-cap cells by using the `-lib_cell` option. For example,

```
icc_shell> add_end_cap -lib_cell MY_END_CAP
```

By default, the command places the specified library cells in their default orientation at both ends of the horizontal cell rows without considering padding, blockages, or keepouts.

- To add end caps to only one end, specify which end by using the `-mode` option.
To add end caps only at the left end, specify the `-mode bottom_left` option. To add end caps only at the right end, specify the `-mode upper_right` option.
- To add vertical end-cap cells, use the `-vertical_cells` option to specify the cells to add as vertical end-cap cells.

The command inserts cells in the specified order and avoids unfilled space at the end of a cell row.

When you add both horizontal and vertical end-cap cells, you can fill the corners where the horizontal and vertical end-cap cells meet by specifying the `-fill_corner` option. The `-fill_corner` option takes effect only when you use it with the `-vertical_cells` option.

- To flip the orientation of the end-cap cells, use the `-mirror` option. The `-mirror` option applies only to horizontal end-cap cells.
- To prevent the command from placing end-cap cells inside padding areas, use the `-respect_padding` option.
- To prevent the command from placing end-cap cells inside blockages, use the `-respect_blockage` option. By default, the command ignores both hard and soft blockages. To respect hard blockages but ignore soft blockages, use the `-ignore_soft_blockage` option with the `-respect_blockage` option. If you specify only the `-ignore_soft_blockage` option, the command issues an error.
- To prevent the command from placing end-cap cells inside keepouts, use the `-respect_keepout` option.

You can also insert end-cap cells at the sides of voltage areas, plan groups, and fixed cells (by default, these objects are ignored when inserting end-cap cells).

- To insert horizontal end-cap cells at both sides of a vertical voltage area boundary, use the `-at_va_boundary` option.
- To insert horizontal end-cap cells at both sides of a vertical plan group boundary, use the `-at_plan_group_boundary` option.
- To treat fixed cells that abut the boundary as macros and insert horizontal end-cap cells next to them, use the `-next_to_fixed` option. Note that this option ignores fixed cells that do not abut the boundary.

To remove end-cap cells from your design, use the `remove_stdcell_filler -end_cap` command.

Inserting Well Fillers

After routing is complete, you can fill small gaps that violate the spacing rule for the well layer with well filler cells. You can fill gaps between cells in the same row or between rows.

To insert well fillers, use the `insert_well_filler` command (or choose Finishing > Insert Well Filler in the GUI).

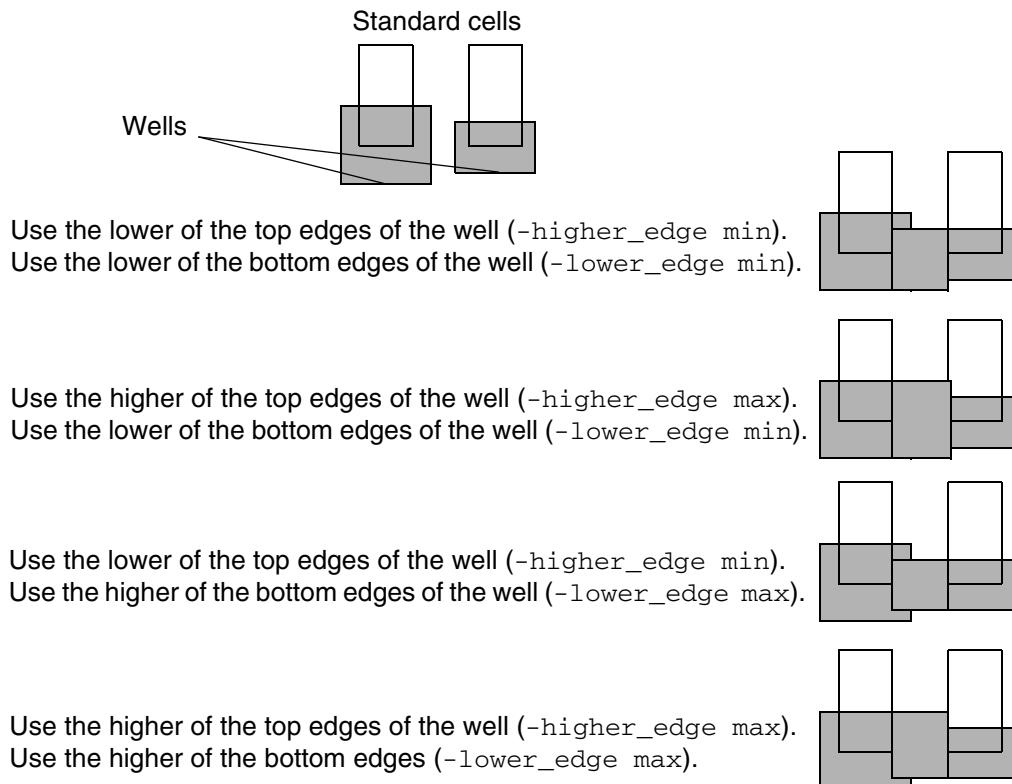
For example, to add well fillers on layer 10 for gaps smaller than 3.0 microns and use the larger top edge and the smaller bottom edge to create the largest fill box if the wells from the two standard cells do not line up, use the following command:

```
icc_shell> insert_well_filler -layer 10 \
    -fill_gaps_smaller_than 3.0 \
    -higher_edge max -lower_edge min
```

By default, only the gaps between standard cells within a row are filled. To fill gaps between rows, use the `-gap_type` option and set it to `tt`, `bb`, `tb`, or `bt`.

The `-higher_edge` and `-lower_edge` options specify how to align the well filler with the wells in the two standard cells when the wells in the cells do not line up. [Figure 7-9](#) shows the alignments and fill boxes that occur as a result of the various settings.

Figure 7-9 Filler Alignment



The `insert_well_filler` command also has options to specify the following:

- Whether to ignore the place and route boundary layer (layer 207) that extends outside a cell
- Whether to respect placement blockages when the `-gap_type` option is used
- The amount of row overlap between the row and the gap when the `-gap_type` option is used
- The minimum and maximum gap sizes that are filled when the `-gap_type` option is used
- Whether to insert a well enclosure instead of well fill and, if so, the enclosure width

To remove well fillers, use the `remove_well_filler` command (or choose Finishing > Remove Well Fillers in the GUI).

Inserting Pad Fillers

After routing is complete, you can fill gaps in the pad ring with instances of pad filler cells. These are dummy pad cells that you can use to control the pad spacing and complement the n-well taps in pads. You should complete the routing process before you add pad filler cells.

To insert pad fillers, use the `insert_pad_filler` command (or choose Finishing > Insert Pad Filler in the GUI).

For example,

```
icc_shell> insert_pad_filler -cell "PFILL_2X PFILL_1X" \
-overlap_cell "PFILL_1X" -voltage_area {V1}
```

This example inserts pad filler cells PFILL_2X and PFILL_1X on the pad ring, with preference for PFILL_2X because it is listed first, inside voltage area V1 only. The PFILL_1X cell is allowed to overlap other pad filler cells to fill gaps that are too small for any pad filler cell.

The command also has options to restrict pad filler insertion to a specified rectangular area or pad ring, to specify pad filler instance naming conventions, and to exclude pad filler insertion from left, right, top, or bottom boundaries.

To remove pad fillers, use the `remove_stdcell_filler -pad` command (or choose Finishing > Remove Fillers in the GUI and select “Pad” in the “Filler type” area). By default, pad fillers are removed for the whole chip. You can optionally specify a bounding box from which to remove pad fillers.

Inserting Metal Fill

After routing, you can fill the empty spaces in the design with metal wires to meet the metal density rules required by most fabrication processes. Before inserting metal fill, the design should be close to meeting timing and have only a very few or no DRC violations.

You insert metal fill by running the `signoff_metal_fill` command (or choosing Finishing > Signoff Metal Fill in the GUI) to invoke the external IC Validator metal fill capability. This command requires an IC Validator license.

After you insert metal fill,

- You can display the added metal fill in the layout view in the GUI.

Before you can view the added metal fill in the GUI, you must open the FILL view. After opening the FILL view, set the View Level to 1 in the View Settings panel (View > Toolbars > View Settings) and toggle on the Datatypes option under the Layers tab. Toggle on the view of the desired fill layer.

- You can do extraction for timing analysis using the real metal fill.

To enable real metal fill extraction, enter the following command:

```
icc_shell> set_extraction_options -real_metalfill_extraction floating
```

The IC Compiler tool uses non-emulation TLUPlus files when performing real metal fill extraction. Before enabling real metal fill extraction, ensure that the design contains metal fill and that you have specified TLUPlus files without any emulation information.

Note:

If you have not yet inserted metal fill, you can perform emulation metal fill extraction by specifying emulation TLUPlus files with the `-max_emulation_tluplus` and `-min_emulation_tluplus` options of the `set_tlu_plus_files` command.

To use the `signoff_metal_fill` command to insert metal fill,

- Set up the validation tool environment.
- Set up the physical signoff options.
- Set up distributed processing.
- Run the `signoff_metal_fill` command.

The following sections describe these tasks.

Setting Up the IC Validator Environment

To run the `signoff_metal_fill` command, you must have an IC Validator license, and you must specify the location of the IC Validator executable by setting the `ICV_HOME_DIR` environment variable. You can set this variable in your `.cshrc` file. To specify the location of the IC Validator executable, use commands similar to those shown in the following example:

```
setenv ICV_HOME_DIR /root_dir/icv  
set path = ($path $ICV_HOME_DIR/bin/AMD.64)
```

You must ensure that the version of the IC Validator executable that you specify is compatible with the IC Compiler version that you are using.

For more information about the IC Validator tool, see the IC Validator documentation, which is available on SolvNet.

The IC Validator tool can generate a compressed hierarchical FILL view file, instead of the default hierarchical FILL view file. The compressed hierarchical FILL view reduces the size of the FILL view file without increasing runtime or memory requirements. To use the compressed hierarchical FILL view file instead of the default hierarchical FILL view file, set the `ICV_ENABLE_GDSREF` environment variable to 1 before invoking the IC Compiler tool. You can also set this variable in your `.cshrc` file.

```
setenv ICV_ENABLE_GDSREF 1
```

Setting the Physical Signoff Options

To prepare for signoff metal fill, use the `set_physical_signoff_options` command to specify the name of the executable (`icv`) and the runset file for metal fill.

For example,

```
icc_shell> set_physical_signoff_options -exec_cmd icv \  
-fill_runset my_fill_runset_file
```

If the layer numbers differ between the Milkyway technology file and the foundry runset file used by the IC Validator tool, you must supply a layer mapping file to map the Milkyway layers to the layers used in the runset file. Use the `-mapfile` option with the `set_physical_signoff_options` command to specify the layer mapping file. For information about creating the layer mapping file, see “[Defining the Layer Mapping Between Milkyway and the Foundry Runset](#)” on page 6-125.

You can report the option settings by using the `report_physical_signoff_options` command.

Setting Up Distributed Processing

By default, the `signoff_metal_fill` command uses a single process to perform metal fill insertion. To reduce the turnaround time for metal fill insertion, you can use distributed processing. To enable distributed processing, you must define the distributed processing configuration by using the `set_host_options` command.

If you have defined more than one distributed processing configuration with the `set_host_options` command, the `signoff_metal_fill` command selects the IC Validator processing method in the following order of priority:

1. Job submission through a user-defined distributed processing script

To enable job submission using your own script with the `set_host_options` command, use the `-submit_command` option to specify the location of your job submission script. For example, to specify a configuration named `custom4` that enables a maximum of four processes using your job submission script, enter the following command:

```
icc_shell> set_host_options -name custom4 -num_processes 4 \
    -submit_command /usr/local/bin/my_submit_command
```

2. Job submission through the Load Sharing Facility (LSF) or the Sun Grid Engine (SGE)

To enable LSF or SGE job submission with the `set_host_options` command, use the `-pool` option to specify the mode and the `-num_processes` option to specify the maximum number of processes.

For example, to specify a configuration named `lsf4` that enables a maximum of four processes using LSF, enter the following command:

```
icc_shell> set_host_options -name lsf4 -pool lsf -num_processes 4
```

To specify a configuration named `grd4` that enables a maximum of four processes using SGE, enter the following command:

```
icc_shell> set_host_options -name grd4 -pool grd -num_processes 4
```

3. Distributed processing on the specified hosts

To enable distributed processing with the `set_host_options` command, specify the hosts to use and use the `-num_processes` option to specify the maximum number of processes on each host. For example, to specify a configuration named `dp4` that enables a maximum of four processes, with a maximum of two processes each on `machineA` and `machineB`, enter the following command:

```
icc_shell> set_host_options -name dp4 -num_processes 2 \
    {machineA machineB}
```

4. Multithreading

To enable multithreading on the current machine with the `set_host_options` command, use the `-max_cores` option to specify the number of threads. For example, to specify a configuration named mt4 that enables a maximum of four threads on the current machine, enter the following command:

```
icc_shell> set_host_options -name mt4 -max_cores 4
```

To ensure that you are using the intended distributed processing configuration, remove the current configurations by using the `remove_host_options` command before defining the distributed processing configuration for the `signoff_metal_fill` command. To report the current distributed processing configurations, use the `report_host_options` command.

For more information about the `set_host_options` command, see “[Enabling Multicore Processing](#)” on page 2-12.

Running the `signoff_metal_fill` Command

Before you run the `signoff_metal_fill` command, the design must be fully routed and have only a very few or no DRC violations, and you must save the most recent revision of the design in a CEL view.

The IC Validator tool reads and operates on the design data in the CEL, FRAM, and FILL views stored on disk, not on the current design in IC Compiler memory. By default, if a cell has both a CEL and FRAM view, IC Validator reads the CEL view and the FILL view, if it exists, and ignores the FRAM view. To read the FRAM view for specific cells instead, use the `-read_fram_view` option. When you use this option, IC Validator ignores the CEL and FILL views for the specified cells. For example, to read the FRAM views for the A1 and A2 macros (and the CEL and FILL views for all other cells), use the following command:

```
icc_shell> signoff_metal_fill -read_fram_view {A1 A2}
```

You can use the `signoff_metal_fill` command to perform the following tasks, which are described in this section:

- [Standard Metal Fill Insertion](#)
- [Timing-Driven Metal Fill Insertion](#)
- [Metal Fill Removal](#)
- [Post-ECO Metal Fill Cleanup](#)

The following sections describe these tasks.

Note:

When you run the `signoff_metal_fill` command, you can specify additional options for the IC Validator command line by using the `-user_defined_options` option. The

string that you specify in this option is added to the command line used to invoke metal fill insertion in the IC Validator tool. The IC Compiler tool does not perform any checking on the specified string.

Standard Metal Fill Insertion

Normal metal fill insertion is the default mode for the `signoff_metal_fill` command. If you run the `signoff_metal_fill` command without any options, it performs the following tasks:

1. Removes existing metal and via fill from the entire design.

You can skip this step and perform incremental metal fill insertion by using the `-append` option (or by selecting “Keep existing metal fills in output view” in the Fill Options tab in the GUI). When you perform incremental metal fill insertion, you must use the default FILL view.

2. Inserts metal and via fill in the empty regions for the whole design using the metal fill mode specified in the runset file, which is either hierarchical or flat.

You can force the use of the flat metal fill mode by using the `-mode flat` option.

Note:

The `signoff_metal_fill` command respects routing blockages defined by the `create_routing_blockage` command and does not insert metal fill in the blockages; however, it does not respect route guides created by the `create_route_guide` command.

If your design contains routing blockages and the blockage layer numbers differ between IC Compiler and the foundry runset file, you must provide a layer mapping file, as described in “[Setting the Physical Signoff Options](#)” on page 7-58.

3. Removes floating via fill. Floating via fill is via fill that does not have both an upper and lower metal enclosure.

You can disable the removal of floating via fill by using the `-remove_floating_via_fills false` option.

4. Stores the inserted metal fill information in the default FILL view.

You can specify a different name for the FILL view by using the `-output_view` option (or by entering the name in the “Output FILL view name” text box in the Fill Options tab in the GUI).

You can restrict the metal fill insertion to specific layers and regions of the design, or, if you have previously run the `signoff_metal_fill` command on the design, to those regions of the design that have been modified since the previous run.

Specifying the Layers for Metal Fill Insertion

By default, the `signoff_metal_fill` command inserts metal fill on all the metal routing layers and via fill on all the via layers. To perform metal fill insertion on a specific set of metal and via layers, specify the layers in the `-select_layers` option (or select them in the list in the “Insert/Remove metal fills in selected layers” area in the GUI).

By default, when you use this option, the `signoff_metal_fill` command

1. Removes all existing metal and via fill from the design.

To keep the existing metal fill on the unspecified layers and to redo metal fill insertion only on the specified layers, use the `-eco` option. If you use the `-eco` option, you must use the default FILL view.

For example, to remove all metal fill on layers M1 and M3 and refill those two layers without affecting the metal fill on other layers, enter the following command:

```
icc_shell> signoff_metal_fill -eco -select_layers {M1 M3}
```

2. Inserts metal and via fill only on the specified layers.
3. Removes floating via fill from the specified via layers and from the via layers associated with the specified metal layers.

For example, if you specify `-select_layers {metal2}`, the tool removes floating via fill from the via1 and via2 layers.

Specifying the Regions for Metal Fill Insertion

By default, the `signoff_metal_fill` command inserts metal and via fill for the whole chip. You can restrict metal fill insertion to one or more regions of the design by specifying regions in which to insert metal fill or by specifying regions in which to prevent metal fill insertion or both.

To restrict metal fill insertion to specific regions of the design, use the `-bounding_boxes` option to specify the coordinates of each region in which to insert metal fill. You can specify multiple areas by specifying the coordinates for each area. If you are using the GUI, identify the regions in which to insert metal fill by selecting “Selected areas” and specifying the coordinates of each bounding box in the “Bounding Boxes” list. To specify the coordinates for a region, either draw the bounding box or enter the x- and y-coordinates of the box. For each region, you can also enable or disable snapping. If snapping is enabled, you can specify that the bounding box should snap to the minimum grid (the default), placement site, routing track, middle routing track, or user grid.

Note:

The bounding box coordinates passed to the IC Validator tool in the `METAL_FILL_SELECT_WINDOW` parameter are enlarged by 1 um to avoid DRC violations on the boundary of the specified regions during metal fill insertion. The actual metal fill insertion occurs within the regions specified by the `-bounding_boxes` option.

In addition, when you use the `-bounding_boxes` option, the `signoff_metal_fill` command always uses flat metal fill mode.

To prevent metal fill insertion in specific regions, use the `-excluded_bounding_boxes` option or select “Excluded areas” and specify the coordinates of each bounding box in the “Bounding Boxes” list in the GUI.

By default, when you use these options, the `signoff_metal_fill` command removes all existing metal and via fill from the design and then inserts metal and via fill only in the specified regions. To redo metal fill insertion only on the specified regions and keep the other existing metal fill, use the `-eco` option. If you use the `-eco` option, you must use the default FILL view.

Note:

When you use the `-eco` option, you must also use one of the following options:

- `-select_layers`,
- `-timing_preserve_nets`,
- `-timing_preserve_setup_slack_threshold`, or
- `-timing_preserve_hold_slack_threshold`.

For example, to remove all metal fill from the design and then fill all empty regions outside the bounding box with corners at (100,150) and (300,200), enter the following command:

```
icc_shell> signoff_metal_fill \
    -excluded_bounding_boxes {{100 150} {300 200}}
```

Performing Metal Fill Insertion Only in Modified Regions

If you have previously run the `signoff_metal_fill` command on a design, you can use the `-auto_eco` option to restrict the metal fill insertion to those regions of the design that have been modified since the last run.

When you use the `-auto_eco` option, the tool automatically determines the modified regions and layers, and then removes the existing metal fill and redoes metal fill insertion only in those locations; all other existing metal fill is retained.

Note:

A change on a metal layer triggers metal fill removal and insertion on the adjacent via layers to ensure that no floating or hanging vias remain when the metal shapes are removed from the ECO area. If a cell instance changes, the tool takes a conservative approach and considers changes on all layers.

For example, to remove the metal fill from the regions modified since the last time you ran the `signoff_metal_fill` command and then fill only those regions, enter the following command:

```
icc_shell> signoff_metal_fill -auto_eco
```

You can use the `-select_layers` option to specify the layers on which to perform metal fill insertion; however, the tool performs metal fill insertion on a specified layer only if it is one of

the automatically detected changed layers. You cannot use the `-bounding_boxes` or `-excluded_bounding_boxes` options to restrict the metal fill insertion regions.

When you use the `-auto_eco` option, you must always use the default FILL view and you cannot use the `-purge` or `-append` option.

Timing-Driven Metal Fill Insertion

Timing-driven metal fill insertion inserts metal and via fill in the specified regions of the design, except around timing-critical nets. By default, the tool does not perform timing-driven metal fill insertion. To enable timing-driven metal fill insertion, you must specify at least one of the following settings:

- Timing-preservation effort level

To set the timing-preservation effort level, set the `-timing_preserve_effort_level` option to `off`, `low`, `medium`, or `high`.

Setting this option to a value other than `off` enables timing-driven metal fill insertion and sets the various timing-related options to preset values. To override these values, use the appropriate options in addition to the `-timing_preserve_effort_level` option.

- Slack thresholds

To set the setup slack threshold, use the `-timing_preserve_setup_slack_threshold` option. To set the hold slack threshold, use the `-timing_preserve_hold_slack_threshold` option.

When you set one or both of these options, the tool identifies the timing-critical nets based on these settings.

If you set the timing-preservation effort level, the tool automatically uses a setup slack threshold based on the specified effort level: 0.02 for low effort, 0.06 for medium effort, or 0.10 for high effort.

If you specify both the setup slack threshold and the timing-preservation effort level, the value you set with the `-timing_preserve_setup_slack_threshold` option overrides the effort-based value.

The unit for the slack threshold setting is the library time unit, whether you explicitly specify the value or it is determined by the timing-preservation effort level.

- Timing-critical nets

To explicitly specify the timing-critical nets, use the `-timing_preserve_nets` option.

By default, the tool uses the minimum spacing values specified in the technology file to determine the spacing requirements between the metal fill and the net shapes. You can control the spacing requirements either explicitly or by setting the timing-preservation effort level. If you specify both the explicit option and the timing-preservation effort level, the

explicit option setting overrides the setting derived from the effort level. [Table 7-6](#) shows the spacing requirements that you can control during timing-driven metal fill insertion, their defaults, and their settings based on the timing-preservation effort level. Note that the defaults apply only when timing-driven metal fill insertion is enabled by explicitly specifying the slack threshold or timing-critical nets without specifying an effort level.

Table 7-6 Spacing Requirements for Timing-Driven Metal Fill Insertion

Description	Default	Timing-preservation effort level		
		low	medium	high
Minimum spacing between metal fill and a timing-critical net on the same layer ¹ (-space_to_critical_nets option, units are the unit length specified in the technology file)	2X	2X	2X	3X
Minimum spacing between metal fill and a timing-critical clock net on the same layer ¹ (-space_to_clock_nets option, units are the unit length specified in the technology file)	2X	3X	3X	4X
Minimum spacing between metal fill and a timing-critical net on an adjacent layer ¹ (-space_to_critical_nets_adjacent_layers option, units are the unit length specified in the technology file)	1X	1X	2X	3X
Metal fill inserted within the minimum spacing around the vertical extension of the net on the layers adjacent to the net (-fill_over_critical_nets option)	false	false	false	false
Nondefault spacing rules honored (-apply_nondefault_rules option)	false	false	true	true

1. Values in the table are specified as a multiple of the minimum spacing defined in the technology file for the layer.

In some cases, the spacing requirements for timing-driven metal fill insertion might cause density errors. If you set a timing-preservation effort level other than `off`, the tool performs density error fixing during timing-driven metal fill insertion. Otherwise, you must use the

`-fix_density_errors true` option to enable density error fixing. When density error fixing is enabled, the tool checks and fixes the following density design rules:

- Minimum density
- Maximum open area
- Density gradient

Note that density error fixing requires that the minimum density and density gradient rules are defined in the technology file. For information about defining metal density rules in the technology file, see the *IC Compiler Technology File and Routing Rules Reference Manual*. For information about defining the maximum open area rule, see “[Defining the Maximum Open Area Rule](#).”

Note:

You can restrict the regions in which to perform timing-driven metal fill insertion, as described in “[Specifying the Regions for Metal Fill Insertion](#)” on page 7-62 and “[Performing Metal Fill Insertion Only in Modified Regions](#)” on page 7-63; however, you cannot specify the layers. When performing timing-driven metal fill insertion, the `signoff_metal_fill` command inserts metal fill on all routing layers (or all changed layers, if you perform metal fill insertion only in the changed regions).

During timing-driven metal fill insertion, the `signoff_metal_fill` command

- Performs timing analysis, including multicorner-multimode analysis, to minimize timing impact.
- Identifies timing-critical nets based on the options you specify.
- Invokes the IC Validator tool to perform metal fill insertion.

By default, when you run timing-driven metal fill insertion, the `signoff_metal_fill` command removes all existing metal and via fill from the design and then inserts metal and via fill in the specified regions, except in the areas around the critical nets.

- To do timing-driven metal fill insertion on the specified regions and keep the other existing metal fill, use the `-eco` option.
- If you have previously run the `signoff_metal_fill` command on the design, use the `-auto_eco` option to do timing-driven metal fill insertion on those regions of the design that have been modified since the previous run.

When you perform timing-driven metal fill insertion, you must always use the default FILL view and you cannot use the `-purge` or `-append` option.

Defining the Maximum Open Area Rule

The maximum open area rule defines the maximum size of a square area that contains no polygons and does not interact with any polygons. This rule is not defined in the technology file. Instead, you must define it in an IC Validator user-defined parameter file. Use the following syntax to define the maximum open area rule for each metal layer:

```
mn_max_open_area_rule = value;
```

where *n* is the metal layer number and *value* is the side length of the square in microns.

To enable checking and fixing of the maximum open area rule, you must use the `-user_defined_options` option with the `signoff_metal_fill` command to specify the name of the user-defined parameter file. For example, to specify a user-defined parameter file named `user_defined_params.rh`, use the following command:

```
icc_shell> signoff_metal_fill -user_defined_options \
{-D USER_DEFINED_PARAM_FILE=user_defined_params.rh}
```

Metal Fill Removal

To remove all metal and via fill from the design, use the `-purge` option (or select “Remove fills” in the GUI). When you use this option, you must use the default FILL view. You can specify the layers on which to remove the metal fill by using the `-select_layers` option. You can specify the regions from which to remove the metal fill by using the `-bounding_boxes` option. No other options are supported with the `-purge` option. For more information about specifying the layers for metal fill removal, see [“Specifying the Layers for Metal Fill Insertion” on page 7-62](#). For more information about specifying the regions for metal fill removal, see [“Specifying the Regions for Metal Fill Insertion” on page 7-62](#).

Post-ECO Metal Fill Cleanup

After you perform metal fill insertion and ECO routing, use the `signoff_metal_fill` command to remove any metal fill that overlaps a net. You can remove the metal fill based on foundry-specific rules or based on minimum spacing rules.

Removing Metal Fill Based on Foundry-Specific Rules

The `signoff_metal_fill` command supports metal fill removal based on the fill spacing rules for several foundry and node combinations. To determine the supported foundry and node combinations, use the `signoff_metal_fill -remove_overlap_by_rules list` command.

To remove the metal fill that overlaps any net based on foundry-specific rules,

1. Prepare the fill removal runset include file.

The fill removal runset include file specifies the foundry's fill spacing rule values. For details about how to prepare this file, see [SolvNet article 035828, "Preparing the Runset Include File for Post-ECO Metal Fill Removal."](#)

2. Specify the fill removal runset include file by using the `-fill_removal_runset_include_file` option with the `set_physical_signoff_options` command.

```
icc_shell> set_physical_signoff_options \
           -fill_removal_runset_include_file file_name
```

3. Perform fill removal by using the `-remove_overlap_by_rules` option with the `signoff_metal_fill` command to specify one of the keywords returned by the `signoff_metal_fill -remove_overlap_by_rules list` command.

```
icc_shell> signoff_metal_fill \
           -remove_overlap_by_rules foundry_node
```

Removing Metal Fill Based on Minimum Spacing

The `signoff_metal_fill` command supports metal fill removal based on either the minimum spacing rules defined in the technology file or user-specified minimum spacing values. When removing metal fill based on the minimum spacing rules defined in the technology file, the tool removes the metal fill around each net such that there is no metal fill within twice the minimum spacing for the layer on which the net occurs. You can either remove the metal fill around all nets or only around specified nets.

To remove the metal fill that overlaps any net based on the minimum spacing rules, use the `-remove_overlap_by_rules min_spacing` option. To use user-defined spacing values instead of the minimum spacing rules, you must also use the `-space_to_critical_nets` option to specify the spacing requirements for each layer.

For example, to use the minimum spacing rules to remove the metal fill that overlaps the existing nets, enter the following command:

```
icc_shell> signoff_metal_fill -remove_overlap_by_rules min_spacing
```

To remove the metal fill that overlaps specific nets based on the minimum spacing rules, use the `-remove_overlap_with_nets nets` option. To use user-defined spacing values instead of the minimum spacing rules, you must also use the `-space_to_critical_nets` option to specify the spacing requirements for each layer.

For example, to remove the metal fill that overlaps the n1 net with a minimum spacing of 0.2 microns on the M1, M2, and M3 metal layers, enter the following command:

```
icc_shell> signoff_metal_fill -remove_overlap_with_nets n1 \
    -space_to_critical_nets {M1 0.2 M2 0.2 M3 0.2}
```

Note:

You can specify the layers on which to remove the metal fill, as described in “[Specifying the Layers for Metal Fill Insertion](#)” on page 7-62; however, you cannot specify the regions. When removing overlapping metal fill, the `signoff_metal_fill` command removes the metal fill around the specified nets.

For example, to use the minimum spacing rules to remove the metal fill on layers M1, M2, and M3 that overlaps existing nets, enter the following command:

```
icc_shell> signoff_metal_fill -remove_overlap_by_rules min_spacing \
    -select_layers {M1 M2 M3}
```

Analyzing the Metal Density

To help you analyze the metal density in your design, the tool can generate a visual metal density map and a visual gradient density map. Before generating these maps, you must calculate the metal density information. To learn about analyzing the metal density, see

- [Calculating the Metal Density Information](#)
- [Displaying the Metal Density Map](#)
- [Displaying the Gradient Density Map](#)

Calculating the Metal Density Information

To calculate the metal density information, run the `report_metal_density` command.

```
icc_shell> report_metal_density
```

This command invokes the IC Validator tool to calculate the metal density and density gradient values for the current design.

Note:

To run the `report_metal_density` command, you must have an IC Validator license, and you must specify the location of the IC Validator executable by setting the `ICV_HOME_DIR` environment variable, as described in “[Setting Up the IC Validator Environment](#)” on page 7-58.

The tool requires grid size information for each metal layer for which to calculate the metal density information. By default, the tool uses the grid sizes defined for each layer by the `windowSize` attribute in the `DensityRule` section of the technology file. If these attributes are not defined in the technology file, or if you want to use different grid sizes, use the `-density_grid_size` and `-gradient_grid_size` options to define the grid size. For

information about defining the metal density and density gradient rules in the technology file, including the `windowSize` attribute, see the *IC Compiler Technology File and Routing Rules Reference Manual*.

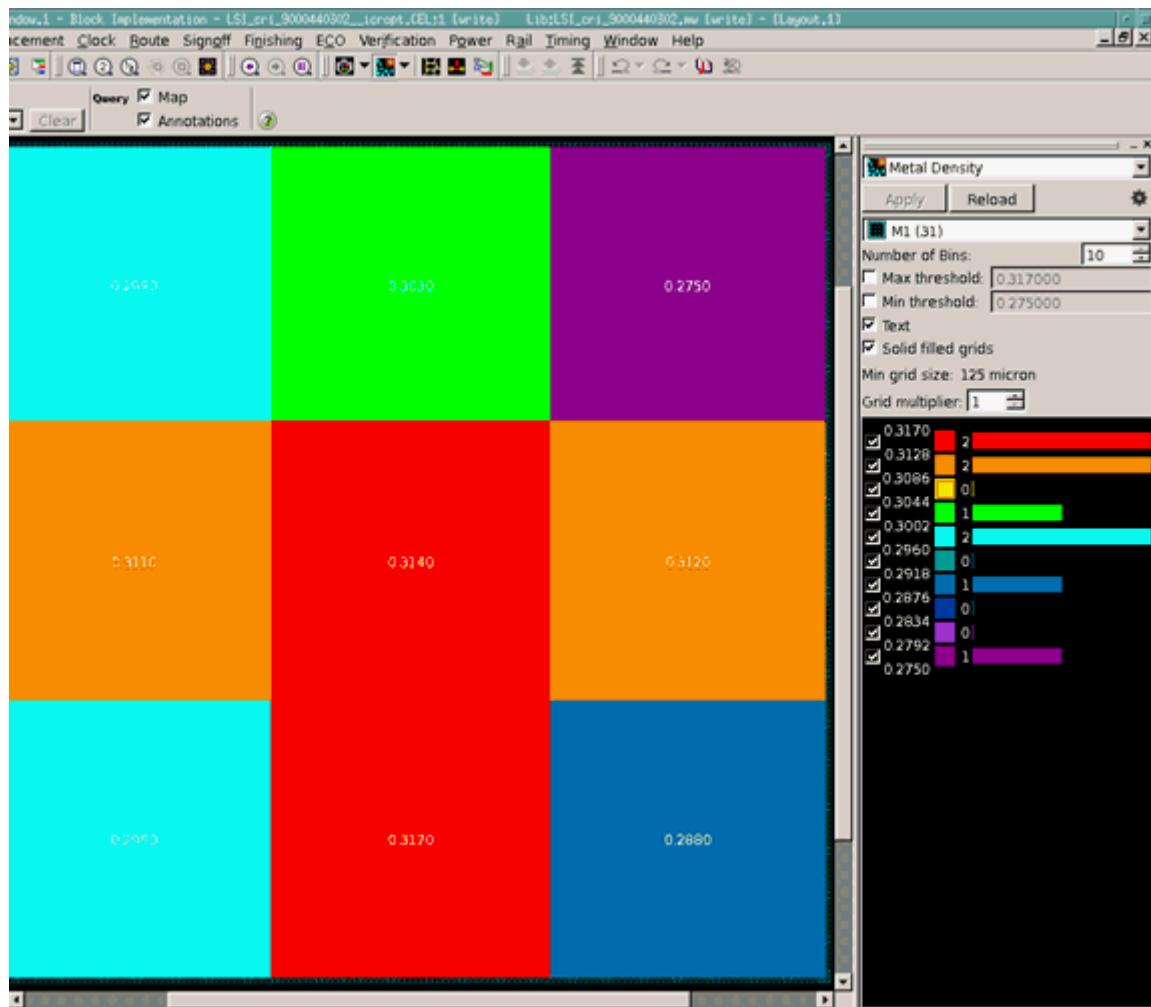
By default, if the timestamp of the FILL view is newer than that of the CEL view, the tool uses the FILL view for the calculations; otherwise, the tool uses the CEL view. To use the FILL view regardless of its timestamp, use the `-ignore_fill_view_time_stamp` option.

After IC Validator calculates the metal density and density gradient values, it annotates the values on the design so that the GUI can access the information to generate the visual displays.

Displaying the Metal Density Map

To display the metal density map, choose Finishing > Metal Density Map in the GUI. If the design database contains metal density information, the tool generates the metal density map based on this information; otherwise, you must click Reload to generate the metal density map. When you click Reload, the tool opens the Calculate Metal Gradient Map dialog box. Specify the appropriate options, as described in “[Calculating the Metal Density Information](#),” and then click OK. When you click OK, the tool generates a new metal density map.

[Figure 7-10](#) shows a metal density map. When the view is zoomed in enough, the density values are displayed inside the grid squares, as shown in [Figure 7-10](#). You can control the display by setting options on the Map Mode panel.

Figure 7-10 Metal Density Map

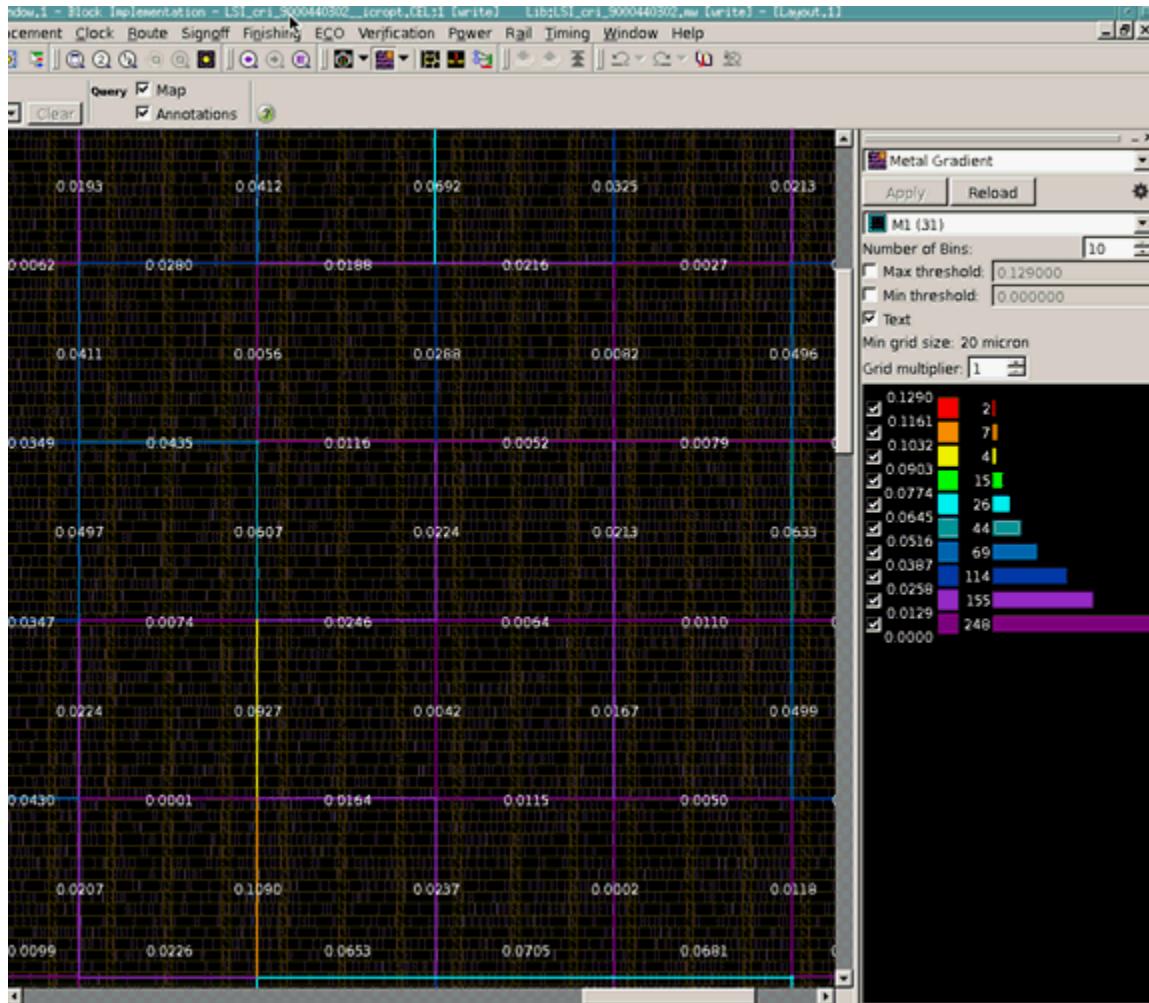
Displaying the Gradient Density Map

To display the gradient density map, choose Finishing > Metal Gradient Map in the GUI. If the design database contains metal density information, the tool generates the gradient density map based on this information; otherwise, you must click Reload to generate the gradient density map. When you click Reload, the tool opens the Calculate Metal Gradient Map dialog box. Specify the appropriate options, as described in “[Calculating the Metal Density Information](#),” and then click OK. When you click OK, the tool generates a new gradient density map.

Figure 7-11 shows a gradient density map. The gradient density map shows the borders between grids highlighted with different colors that represent the different gradient levels. When the view is zoomed in enough, the gradient values are displayed inside the grid

squares, as shown in [Figure 7-11](#). You can control the display by setting options on the Map Mode panel.

Figure 7-11 Gradient Density Map



Part II: Advanced IC Compiler Features

8

Signal Integrity

Signal integrity is the ability of an electrical signal to carry information reliably and to resist the effects of high-frequency electromagnetic interference from nearby signals. This chapter describes how to use the IC Compiler tool to analyze and correct signal integrity problems.

Crosstalk is the undesirable electrical interaction between two or more physically adjacent nets due to capacitive coupling. Crosstalk can lead to crosstalk-induced delay changes or static noise that causes logic errors.

Electromigration is the permanent physical movement of metal in thin wire connections resulting from the displacement of metal ions by flowing electrons. Electromigration can lead to shorts and opens in wire connections, causing functional failure of the device. The problem is more severe in modern technologies due to smaller wire widths and increased current densities.

The IC Compiler tool supports signal integrity analysis and optimization in either a flat flow or a hierarchical flow. When you create a hierarchical model for use in a hierarchical signal integrity flow, the signal integrity information is automatically included by the `create_block_abstraction` command. Signal integrity analysis and optimization also support multivoltage and multimode-multicorner designs.

This chapter contains the following sections:

- [Analyzing and Reducing Crosstalk](#)
- [Analyzing and Reducing Signal Electromigration](#)

Analyzing and Reducing Crosstalk

Crosstalk is the undesirable electrical interaction between two or more physically adjacent nets due to capacitive coupling. The two major effects of crosstalk are crosstalk-induced delay and static noise.

- Crosstalk can affect signal delays by changing the times at which signal transitions occur. That is, it can either speed up or slow down the transition time of the victim net. These changes occur when both the victim and the aggressor nets are switching at the same time. When both are switching in the same direction, the transition time of the victim net decreases and delay is reduced. When the victim and aggressor nets are switching in opposite directions, the transition time of the victim net increases and delay is increased. Therefore crosstalk can affect both setup and hold checking. The noise on the victim net is referred to as switching noise.
- Static noise occurs when the victim net is in a steady state of either a high or low, and the aggressor net is switching. This causes a glitch on the victim net. If a glitch is of sufficient height and width, and if it occurs at the input of a latching structure during a latching operation (possibly as the result of an earlier glitch propagated through combinational logic), a functional failure results.

The IC Compiler tool uses crosstalk prevention techniques during track assignment. After you perform detail routing, the tool performs crosstalk-induced noise and delay analysis to identify any remaining violations, and it fixes these violations during the postroute optimization phases.

- During timing- and crosstalk-driven track assignment, the tool minimizes the crosstalk effects by assigning long, parallel nets to nonadjacent tracks. It runs a simplified noise analysis to make sure the noise level from aggressor nets is minimized.
- After detail routing is complete, the tool repairs the remaining problems, first by analyzing the coupling capacitance effects of the circuit and then by crosstalk removal with postroute optimization.

In crosstalk analysis, for each net, the tool extracts parasitic capacitance to the ground as well as any coupling capacitance with neighboring wires. The tool also invokes static timing analysis, using the extracted parasitics to obtain the transition times as well as the arrival time windows of every signal net in the design.

This section contains the following information about the crosstalk capabilities supported by the IC Compiler tool:

- [Setting the Signal Integrity Options](#)
- [Preventing Crosstalk During Placement](#)
- [Preventing Crosstalk During Clock Tree Synthesis](#)

- Preventing and Fixing Crosstalk During Routing
- Analyzing Crosstalk
- Example Scripts

Setting the Signal Integrity Options

The signal integrity options affect the analysis of crosstalk delay and static noise by the `report_timing` and `report_noise` commands and the optimization of crosstalk delay and static noise by the `route_opt` command.

You set the signal integrity options by using the `set_si_options` command (or by choosing Timing > Set SI Options in the GUI). [Table 8-1](#) describes the GUI objects and command-line options used to set the options for various signal integrity tasks.

Table 8-1 Signal Integrity Options

GUI object	Command option	Default
Crosstalk prevention options		
“Enable xtalk prevention” check box	<code>-route_xtalk_prevention</code>	false
“Prevention threshold” box	<code>-route_xtalk_prevention_threshold</code>	0.35
Crosstalk optimization options		
“Delta delay” check box	<code>-delta_delay</code>	false
“Min delta delay” check box	<code>-min_delta_delay</code>	false
“Static noise” check box	<code>-static_noise</code>	false
“Static noise threshold (voltage)” fields	<code>-static_noise_threshold_above_low</code> <code>-static_noise_threshold_below_high</code>	0.35 0.35

Table 8-1 Signal Integrity Options (Continued)

GUI object	Command option	Default
Crosstalk analysis options		
“Delta delay” check box	-delta_delay	false
“Min delta delay” check box	-min_delta_delay	false
“Static noise” check box	-static_noise	false
“Static noise threshold (voltage) fields	-static_noise_threshold_above_low -static_noise_threshold_below_high	0.35 0.35
“Timing window” check box	-timing_window	false

To report the current signal integrity option settings, run the `report_si_options` command.

Preventing Crosstalk During Placement

To prevent crosstalk at the placement phase, do the following:

- Minimize congestion.
Reducing congestion improves the routability of the design and adds more resources for extra spacing requirements during crosstalk-driven global route and track assignment. Fixing crosstalk on a congested design is very difficult.
- Have good timing closure before crosstalk fixing. If necessary, add more margin before crosstalk fixing.
- Use area recovery on noncritical paths during `place_opt` to reduce congestion and improve routability. Set the `physopt_area_critical_range` variable to a value that is about 15 percent of the clock period to avoid downsizing paths close to zero slack.
- Help prevent crosstalk by controlling the maximum transition constraint defined on the design. The maximum transition constraint is technology and library dependent. You need to find the best tradeoff between a low maximum transition constraint and congestion. The maximum transition constraint can be relaxed during postroute optimization.

- Use the maximum net length constraint in the IC Compiler tool to minimize the crosstalk effect by preventing very long wires. You must determine the best tradeoff between a low `max_net_length` value and congestion. For a design without macros or only small macros, the `max_net_length` value might be 1000.0; for a design with large macros or high utilization, the `max_net_length` value might be 2000.0 or 3000.0.

Preventing Crosstalk During Clock Tree Synthesis

Because clock nets are typically high-frequency nets, they are often strong aggressor nets. You can prevent crosstalk by shielding clock nets with ground wires.

The IC Compiler tool handles clock shielding with nondefault routing rules. For information about the clock shielding flow, see “[Shielding Clock Nets](#)” on page 5-55.

Preventing and Fixing Crosstalk During Routing

During the `route_opt` command, the IC Compiler tool can perform the following signal integrity tasks:

- Preventing crosstalk (during global routing and track assignment)
- Fixing crosstalk violations during postroute optimization

By default, these capabilities are disabled. To enable these capabilities, set the appropriate signal integrity options and then run the `route_opt` command with the `-xtalk_reduction` or `-only_xtalk_reduction` option.

The `set_route_opt_zrt_crosstalk_options` command sets several options that affect crosstalk reduction. For more information, see “[Setting Crosstalk Reduction Options](#)” on page 6-66.

Preventing Crosstalk

The `route_opt` command can prevent crosstalk in the following ways:

- Crosstalk prevention during track assignment

Enable crosstalk prevention by running `set_si_options -route_xtalk_prevention true` and using the `-xtalk_reduction` option when you run `route_opt`.

You can set the crosstalk prevention threshold voltage by using the `-route_xtalk_prevention_threshold` option with the `set_si_options` command. The lower the threshold voltage, the more that the router tries to prevent crosstalk. The default threshold is 0.45 volts, which could be too relaxed. If your design has crosstalk violations, you should use a lower crosstalk prevention threshold during track assignment, in the range of 0.25 to 0.35 volts.

When you enable crosstalk prevention, the `route_opt` command 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 potential noise exceeding the noise threshold.

- Timing-driven global routing

Enable timing-driven global routing by using the `set_route_zrt_global_options -timing_driven true` command (or by choosing Route > Routing Setup > Set Global Route Options in the GUI and selecting “Timing driven” in the Run control tab).

- Timing-driven track assignment

Enable timing-driven track assignment by using the `set_route_zrt_track_options -timing_driven true` command (or by choosing Route > Routing Setup > Set Track Assign Options in the GUI and selecting “Timing driver”).

Fixing Crosstalk Violations During Postroute Optimization

For best results, use the crosstalk prevention process in conjunction with crosstalk fixing. When crosstalk is prevented in the placement stage and then in the track assignment stage, the routed design has a smaller number of crosstalk violations. Consequently crosstalk fixing in the postroute optimization stage is more effective and has a shorter runtime.

Some crosstalk violations might still remain unfixed after you run the noise avoidance processes that occur in the track assignment phase. The IC Compiler tool fixes the remaining crosstalk violations in the postroute optimization stage.

The effort level you specify for the `route_opt` command determines the effort level the tool uses to fix crosstalk violations during postroute optimization.

- If you use the `-effort medium` option, the tool performs default crosstalk reduction.
- If you use the `-effort high` option, the tool performs high-effort crosstalk reduction, which gives better QoR than the default setting. However, it increases the runtime.

During postroute optimization, the `route_opt` command performs the following crosstalk optimizations based on the signal integrity options you set previously with the `set_si_options` command:

- Optimization with crosstalk delta delay

To optimize timing with crosstalk delta delay, set the `-delta_delay` option to `true`.

- Hold time optimization with crosstalk delta delay

By default, the `route_opt` command performs both setup and hold fixing when the `-delta_delay` option is set to `true`. To perform setup fixing only, set the `-min_delta_delay` option to `false`.

- Static noise

To fix static noise violations, set the `-delta_delay` and `-static_noise` options to `true`.

The default threshold (both high and low) for static noise is 0.35 volts. You can override these threshold values with the `-static_noise_threshold_above_low` and `-static_noise_threshold_below_high` options of the `set_si_options` command. Specify a threshold value as a fraction of the supply voltage, not as an absolute voltage.

For example, if you specify a threshold value of 0.3, as shown in the following example, this means 30 percent of the voltage.

```
icc_shell> set_si_options -static_noise true \
-static_noise_threshold_above_low 0.3 \
-static_noise_threshold_below_high 0.3
```

Check the calculated threshold values by using the `report_si_options` command.

```
icc_shell> report_si_options
```

```
Static Noise Thresholds:
0.3 (0.49V) above low
0.3 (0.49V) below high
```

By default, the tool uses only NLDM noise modeling information. To use CCS noise modeling information, set the `rc_noise_model_mode` variable to `true`. The default is `basic`.

You can perform postroute crosstalk reduction on specific nets by using the `focal_opt -xtalk_reduction_nets` command. For more information, see “[Performing Focal Optimization](#)” on page 6-104.

Analyzing Crosstalk

After running `route_opt`, you can perform crosstalk analysis on your design. The following sections describe how to prepare for and run crosstalk analysis.

Preparing for Crosstalk Analysis

Before you run crosstalk analysis or timing analysis to report static noise and crosstalk-induced delay, you must specify the following:

- The logic libraries, as described in “[Setting Up the Logic Libraries](#)” on page 3-3, to ensure that you are using the correct timing and noise models
- The Arnoldi delay calculation algorithm, as described in “[Selecting the Delay Calculation Method](#)” on page 3-49
- The types of crosstalk analysis to perform

You can select delta delay (`set_si_options -delta_delay true`) and static noise (`set_si_options -static_noise true`).

- The electrical filtering parameters

Electrical filtering eliminates aggressor nets from analysis based on the size of the voltage bump induced on the victim net by the aggressor net. These filters are used by crosstalk analysis to reduce runtime and for consistency with the PrimeTime SI tool.

If the default electrical filtering parameters do not meet your requirements, set the following variables:

- `si_filter_accum_aggr_noise_peak_ratio` (default is 0.03)
- `si_filter_per_aggr_noise_peak_ratio` (default is 0.01)

- The global noise thresholds

A crosstalk violation occurs when the crosstalk-induced noise voltage exceeds the specified noise threshold voltage (at or above which a false transition is likely to be triggered). Crosstalk analysis uses the noise threshold to report static noise violations. By default, both the above-low and below-high thresholds are set to 0.35 volts. You can override these thresholds by running the `set_si_options` command with the `-static_noise_threshold_above_low` and `-static_noise_threshold_below_high` options. Specify a threshold value as a fraction of the supply voltage, not as an absolute voltage.

- Noise modeling

To perform static noise analysis by using the `report_noise` command, the logic library must contain information about the drive characteristics and noise immunity of the pins of library cells. To check the noise modeling information, use the `check_noise` command. This command checks for the presence and validity of noise models on the driver and load pins of the design and reports any pins that are not constrained for noise.

If noise modeling information is missing or you want to change the noise models, you can do so by using the following commands:

```
set_steady_state_resistance  
set_noise_immunity_curve  
set_noise_margin  
set_noise_lib_pin
```

The `set_steady_state_resistance` command specifies the drive resistance at an output of a library cell or at an output port of design, which affects the size of noise bumps at that output in the presence of crosstalk noise. For example,

```
icc_shell> set_steady_state_resistance -low 0.042 my_lib/AN2/Z
```

The `set_noise_immunity_curve` command specifies the noise immunity characteristics at the input of a library cell or at an input port in terms of the noise bump width, height, and area that can be tolerated. For example,

```
icc_shell> set_noise_immunity_curve -low \
-width 0.00 -height 0.58 -area 0.0064 my_lib/AN2/A
```

The `set_noise_margin` command specifies the noise immunity characteristics at the input of a library cell or at an input port in terms of the noise bump height that can be tolerated. For example,

```
icc_shell> set_noise_margin -low 0.4 my_lib/AN2/A
```

The `set_noise_lib_pin` command specifies the noise characteristics of a pin as the same as those of an existing library pin. This is useful when a pin does not have noise information defined, but the pin is known to have the same or similar characteristics as another pin that has defined noise characteristics. For example,

```
icc_shell> set_noise_lib_pin \[get_pins macro_cell/A] \
ccs_noise_lib/inv/IN
```

This example causes the pin `macro_cell/A` to have the noise characteristics of library pin `ccs_noise_lib/inv/IN`.

For background information about noise modeling, see the *PrimeTime User Guide*, available on SolvNet in the PrimeTime Suite documentation set.

- Whether to include timing windows

The effect of crosstalk between two nets depends largely on the overlap of timing windows between the two nets. For example, if the aggressor switches when the victim is in steady state, it induces a noise bump on the victim. If the aggressor switches when the victim is also switching, it can cause the victim to switch faster or slower. If more than one aggressor switches in the same timing window, the effect of the two is the accumulated effect of each on the victim net.

The edges of a timing window at a particular pin are calculated as the earliest and latest possible arrival times of the signal at that point, considering multiple paths to that point. If there is only one path to that endpoint, the timing window is very narrow.

To include timing windows, use the `-timing_window true` option together with the `-delta_delay` or `-static_noise` option of the `set_si_options` command. Using timing windows increases the accuracy of crosstalk analysis at the cost of more runtime.

For a multicorner-multimode design, timing windows are included during crosstalk analysis only if a scenario is enabled for both maximum (setup) and minimum (hold) delay optimization and analysis. For more information about enabling a scenario for maximum and minimum delay optimization and analysis, see “[Enabling and Disabling Scenarios for Specific Tasks](#)” on page 3-37.

Running Crosstalk Analysis

You can perform crosstalk analysis from the command line or in the GUI.

When you perform crosstalk analysis, ensure that

- The input transition time for all the cells in the design is reasonable (less than 1 ns). This is to avoid pessimism in the calculation of peak noise and crosstalk-induced delta delay.
- You use the `set_driving_cell` command to specify the driving cell instead of specifying the input transition for each input pin. The boundary conditions you specify for the input pins determine the driving resistance calculated for the nets assigned to the pins. Using the `set_driving_cell` approach enables the IC Compiler tool to estimate the driving resistance more accurately and to correlate better with the PrimeTime SI tool.

Analyzing Crosstalk-Induced Delay Shift

Crosstalk affects the timing of the design when any of the aggressors switch before the transition of the victim signal has finished. When the aggressor and the victim are switching in the same direction, the victim driver can expend less effort on charging the coupling capacitance and more on increasing the switching speed. This can lead to hold time problems. Conversely, when the aggressor and victim switch in opposite directions, the transition time of the victim is longer because of a larger effective capacitance, and could result in a setup time problem. Therefore, to accurately analyze the timing behavior of a circuit, it is necessary to take into consideration the effects of crosstalk.

The IC Compiler tool analyzes the impact on interconnect timing in terms of propagation delay and transition time. The delay shift of a victim net depends on the arrival time window of its aggressors and also affects the net timing windows. During crosstalk-induced delay, IC Compiler takes into account the timing windows of the victim and aggressor nets. For an accurate idea of the true delay shift, you must do iterations between the timing propagation in the static timing analyzer and the net delay calculation, with crosstalk considered.

To display the crosstalk delta delay, use one of the following methods:

- Use the `report_timing -crosstalk_delta` command.
- Use the delta delay visual mode by following these steps:
 1. Choose Timing > Delta Delay Visual Mode.
The Visual Mode panel appears.
 2. Click Reload.
The Delta Delay Visual Mode dialog box appears.
 3. Specify the analysis options.
 4. Click OK.

To get detailed information about the crosstalk calculations for a particular victim net, use the `report_delay_calculation -crosstalk` command.

Analyzing Static Noise

For static noise, crosstalk analysis reports noise violations (above-low and below-high) that are caused by aggressor net transitions. Above-low noise violations occur when the victim net is at the steady state of logic 0 and the aggressor net is switching from logic 0 to logic 1. Similarly, below-high noise violations occur when the victim net is at a steady state of logic 1 and the aggressor is switching from logic 1 to logic 0. When the above-low (rise) and below-high (fall) noise violations exceed the logic thresholds of the technology, they can cause logic failures.

To display static noise violations, use one of the following methods:

- Use the `report_noise` command.
- Use the `get_si_xtalk_bumps` command.
- Use the static noise visual mode by following these steps:
 1. Choose Timing > Static Noise Visual Mode.
The Visual Mode panel appears.
 2. Click Reload.
The Static Noise Visual Mode dialog box appears.
 3. Specify the analysis options.
 4. Click OK.
- Display the Static Noise Analysis window by following these steps:
 1. Choose Timing > Static Noise Analysis.
The Static Noise Analysis dialog box appears.
 2. Specify the analysis options.
 3. Click OK.

The Static Noise Analysis window provides detailed information about the victim and aggressor nets in your design. It also provides the ability to cross-highlight the aggressor and victim nets in the layout window.

To get detailed information about the noise calculations, use the `report_noise_calculation` command.

Example Scripts

[Example 8-1](#) shows a script for running the flat signal integrity flow (or the top-level hierarchical signal integrity flow).

Example 8-1 Flat Signal Integrity Flow

```
open_mw_lib design
open_mw_cel block
place_opt
clock_opt
set_si_options -delta_delay true -static_noise true
route_opt -xtalk_reduction
save_mw_cel
```

[Example 8-2](#) shows a script for running the block-level hierarchical signal integrity flow with block abstractions.

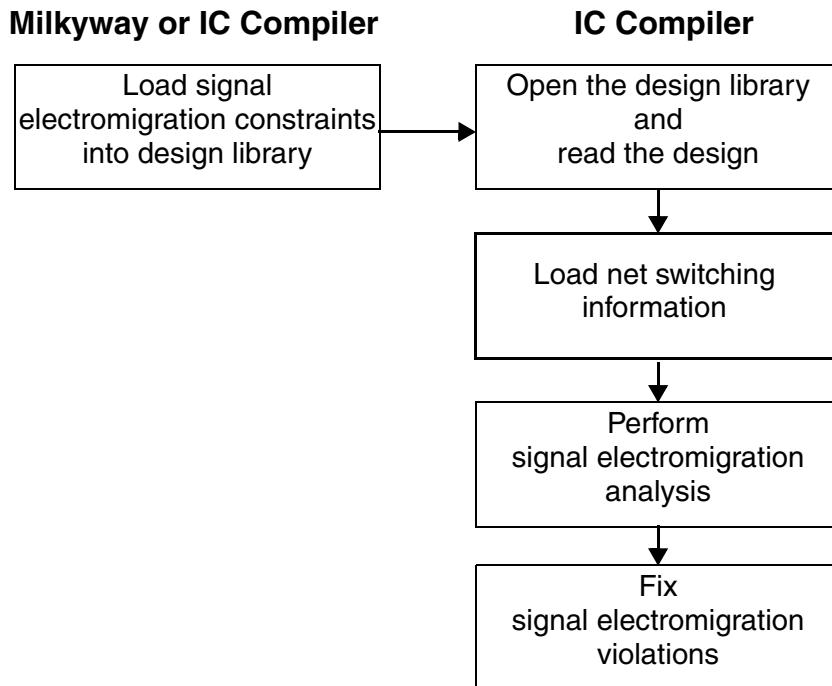
Example 8-2 Block-Level Hierarchical Signal Integrity Flow With Block Abstractions

```
open_mw_lib design
open_mw_cel block
place_opt
clock_opt
set_si_options -delta_delay true -static_noise true
route_opt -xtalk_reduction
create_block_abstraction
save_mw_cel
```

Analyzing and Reducing Signal Electromigration

Signal electromigration problems result from an increase in current density caused by the use of smaller line widths and higher operational speeds in IC designs. Electromigration can lead to shorts or opens due to metal ion displacement caused by the flow of electrons. You can use the IC Compiler tool to analyze and correct electromigration in your design.

[Figure 8-1](#) shows the signal electromigration flow described in the following sections.

Figure 8-1 Signal Electromigration Flow

The following IC Compiler script example shows the signal electromigration flow:

```

# open design library and design cell
open_mw_lib design_library
open_mw_cel design_cell
read_sdc design.sdc

# load switching information
read_saif -input switching_info.saif

# optional override of default switching activity
set_switching_activity -toggle_rate 1.0 \
    -static_probability 0.5 [get_nets net_name]

# perform signal electromigration analysis
report_signal_em -violated -verbose > signal_em.rpt

# fix signal electromigration violations
fix_signal_em

```

Loading Signal Electromigration Constraints

The first step is to load the signal electromigration constraints into the design library. You can do this either in the Milkyway or IC Compiler tool.

Note:

For advanced-node designs, the signal electromigration constraints are in TLUPlus format. For information about loading these constraints from the electromigration TLUPlus file, see the *IC Compiler Advanced Geometries User Guide*.

In the Milkyway Environment, to load the signal electromigration constraints in .plib format into the design library, use a command similar to the following:

```
Milkyway> read_plib -lib_name my_designlib \
    -tech_plib_files signal_em_constraintsplib \
    -overwrite_existing_tech
```

If the electromigration constraints are in Advanced Library Format (ALF), you can read them into the Milkyway library with the an IC Compiler command similar to the following:

```
icc_shell> set_mw_technology_file -alf my_alf_file.alf my_designlib
```

The existing technology data in the current design library is not affected. However, if the existing design library has electromigration data, that information is completely replaced by the new information read from the ALF file.

The `set_mw_technology_file` command accepts only one of the options `-technology`, `-plib`, or `-alf` in any one usage of the command. To load in a technology file and a separate ALF file, two commands are required. For example,

```
icc_shell> set_mw_technology_file -technology mytech.tf my_designlib
icc_shell> set_mw_technology_file -alf my_alf.alf my_designlib
```

To load in a .plib file and a separate ALF file, two commands are required. For example,

```
icc_shell> set_mw_technology_file -plib myplibplib my_designlib
icc_shell> set_mw_technology_file -alf my_alf.alf my_designlib
```

In this example, if the .plib file contains electromigration rules, those rules are completely replaced by the ones read in from the .alf file.

If the signal electromigration constraints are not defined in the design library, you can get an updated incremental .plib file containing the electromigration data from your vendor, and then read it into the Milkyway database with the following IC Compiler command:

```
icc_shell> set_mw_technology_file -plib plib_file_name design_lib_name
```

Verifying the Electromigration Constraints

To perform a check on the signal electromigration constraints in the library, use the following commands:

```
icc_shell> set_check_library_options -signal_em  
1  
icc_shell> check_library  
  
#BEGIN_CHECK_LIBRARY  
...  
#BEGIN_CHECK_SIGNALEM  
  
... (signal electromigration checking results here) ...  
  
#END_CHECK_SIGNALEM  
...  
#END_CHECK_LIBRARY
```

Verifying the Current Unit

To verify that the current unit is defined in the design library, use the `report_units` command.

```
icc_shell> report_units
```

If the current units are not defined, add the following lines to the header section of the technology file, and then update the design library, as described in “[Changing Physical Library Information](#)” on page 3-10:

```
unitCurrentName = "mA"  
currentPrecision = 10000
```

Validating the Design

To perform signal electromigration analysis, the design must have timing constraints and parasitic information. If your design does not yet have timing constraints, see “[Setting Timing Constraints](#)” on page 3-47. If your design does not yet have parasitic information, either back-annotate the data, as described in “[Back-Annotating Delay or Parasitic Data](#)” on page 3-54; or extract the data, as described in “[Postroute RC Extraction](#)” on page 6-146.

Loading the Net Switching Information

You must define net switching information to be able to perform accurate electromigration analysis. (The more frequently a net switches, the more susceptible it is to electromigration.) To load the net switching activity information, either read in a SAIF file with the `read_saif`

command or annotate the switching activity information on the design nets with the `set_switching_activity` command. If you do both, the annotated switching activity has priority.

Important:

You must reload the net switching information every time the netlist changes. If you do load the switching activity, the IC Compiler tool generates a warning message and uses $1/time_unit$ as the net switching frequency, where $time_unit$ is the main library time unit.

Reading a SAIF File

To read in a SAIF file, use the `read_saif` command.

```
icc_shell> read_saif -input saif_file
```

[Example 8-3](#) shows an example SAIF file. For more information about the SAIF file, see the *Power Compiler User Guide*.

Example 8-3 Example SAIF File

```
(SAIFILE
  (SAIFVERSION "2.0")
  (DIRECTION "backward")
  (DATE "Thu Dec 12 15:44:56 2002")
  (VENDOR "Synopsys, Inc")
  (PROGRAM_NAME "Power Compiler PLI")
  (VERSION "3.4")
  (DIVIDER / )
  (TIMESCALE 1 ns)
  (DURATION 22500.00)
  (INSTANCE tb
    (INSTANCE u_A7S
      (NET
        (U_TAP_DBG\ /U_DBG\ /address_d_25_
          (T0 22490) (T1 0) (TX 10)
          (TC 0) (IG 0)
        )))))
))
```

Annotating the Switching Activity

To annotate the switching activity on design nets, use the `set_switching_activity` command with the `-static_probability`, `-toggle_rate`, and `-period` options. After annotating the switching activity, use the `propagate_switching_activity` command to propagate the static probability and toggle rate information to unannotated design objects.

A simple switching activity definition consists of the static probability and the toggle rate. The static probability is the probability that the value of the design object has logic value 1. The toggle rate is the rate at which the design object switches between logic values 0 and 1.

The following example shows how to specify that the value of the net1 net is logic 1 for 20 percent of the time, and that it transitions between logic values 0 and 1 an average of 10 times in 1,000 time units.

```
icc_shell> set_switching_activity [get_net net1] \
    -static_probability 0.2 -toggle_rate 10 -period 1000
icc_shell> propagate_switching_activity
```

The time unit used for the toggle rate is the main library time unit. The `-period` option is optional, and a default of 1 is used when it is not specified.

Performing Signal Electromigration Analysis

After you prepare the design library and load the net switching activity information, you can perform signal electromigration analysis. You use signal electromigration analysis at the postroute stage to fix electromigration violations.

Use the `report_signal_em` command to perform signal electromigration analysis for each net by calculating the current on every edge and comparing this data with the library-defined current-density thresholds. For example,

```
icc_shell> report_signal_em -violated -verbose
```

The `report_signal_em` command performs a timing update, if needed, and analyzes all nets for electromigration. The `-verbose` option causes the report to show detailed electromigration analysis information, which usually produces a very large report when used by itself. To get a report with a reasonable size, use the `-violated` option as well, which limits the report to only the nets with electromigration violations.

To use the IC Compiler GUI to perform electromigration analysis, choose Route > Signal Electromigration > Analyze Signal EM. The dialog box offers the same options as the `report_signal_em` command.

The electromigration analysis produces an error view, which you can read into the IC Compiler GUI to graphically view the locations of the violations. To see the error view in the GUI, choose Verification > Error Browser, which opens the Open Error View dialog box. Toggle on the Signal EM check box and enter the name of the error view, then click OK. The default name of the error view is `current_cell_name_signalem.err`. The `-error_view` option of the `report_signal_em` command lets you explicitly specify the name of the error view created by electromigration analysis.

You can also view a color-coded analysis map showing the local amount of current for a net. To display the map, choose Route > Signal Electromigration > Signal EM Analysis Map. In the Signal EM Analysis window, choose the type of electromigration value (RMS, Avg, or Peak), the number of bins, and the value range. Click Reload to read the electromigration data for a specified net. Then click Apply to display the analysis map and a histogram of current values for the net.

You can specify settings for electromigration analysis and fixing by using the `set_em_options` command.

Table 8-2 set_em_options Command Options

To do this	Use this option
Specify a healing factor.	<code>-healing_factor</code>
Specify the type of constraint rules to use.	<code>-violation_rule_types</code> <code>mean abs_avg rms peak auto</code>
Specify the type of library (maximum or minimum) to use for electromigration calculation.	<code>-max</code> <code>-min</code>
Specify the analysis effort.	<code>-analysis_effort low medium</code>
Specify a minimum duty ratio for computing the peak current.	<code>-min_duty_ratio</code>
Specify the rating factor for the RMS constraints considering the routing wire density.	<code>-rms_factor_for_line_number</code>
Specify the rating factor for the RMS constraints considering the average temperature rise.	<code>-rms_factor_for_avg_temperature</code>
Specify the delta temperature for the RMS constraints.	<code>-delta_temperature</code>
Specify the duration condition for the peak rule of signal electromigration.	<code>-duration_condition_for_peak</code>

To view the current settings, use the `report_em_options` command.

When you perform signal electromigration analysis, the tool computes three current values:

- Average

The average current is calculated by using the following equation:

$$I_{avg} = \frac{\alpha}{2} \int_0^{T/2} I_p(t) dt - \gamma \frac{\alpha}{2} \int_0^{T/2} I_n(t) dt$$

where alpha (α) is the toggle rate, T is the clock period, and gamma (γ) is the healing factor (as specified by the `-healing_factor` option).

- Root mean square

The root mean square current is calculated by using the following equation:

$$I_{rms} = \sqrt{\frac{\alpha}{2} \left[\int_0^{T/2} I_p^2(t) dt + \int_0^{T/2} I_n^2(t) dt \right]}$$

where alpha (α) is the toggle rate and T is the clock period.

- Peak

The peak current is calculated by using the following equation:

$$I_{peak} = MAX(MAX(I_p) \cdot \sqrt{D_p}, MAX(I_n) \cdot \sqrt{D_n})$$

where D is the duty cycle, defined as 50 percent of the peak width multiplied by half of the switching activity, as shown in the following equation:

$$D_p = Width(I_p) \cdot (\alpha/2)$$

$$D_n = Width(I_n) \cdot (\alpha/2)$$

where alpha (α) is the toggle rate. The subscripts p and n denote the rise and fall transitions, respectively.

The default report generated by the `report_signal_em` command lists the number of nets with electromigration violations and the types of constraints violated (mean, absolute average, RMS, or peak). If you use the `-verbose` option, the report displays detailed information about the violations, as shown in [Example 8-4](#).

Example 8-4 Verbose Electromigration Report

```
icc_shell> report_signal_em -violated -verbose
*****
Units:
...
*****
Net name:          net_577
Switching activity: 1.500000
Driver pin name (Transitions rise_min fall_min rise_max fall_max):
                buff_a/z (0.024464 0.0242115 0.0244641 0.0242115)
Violation:        RMS



| LAYER<br>NAME | METAL<br>VIA | LAYER<br>ID | CENTER<br>COORDINATE | WIDTH<br>VIA CUT | MODE | CURRENT<br>Mean(Limit, Violated) | RMS(Limit, Violated)  | Peak(Limit, Violated) |
|---------------|--------------|-------------|----------------------|------------------|------|----------------------------------|-----------------------|-----------------------|
| M4            | Metal        | 34          | [ 293.97, 1857.80]   | 0.07             | min  | 6.66e-16 (2.57e-01,-)            | 6.70e-01 (5.23e-01,v) | 5.67e-01 (8.68e-01,-) |
| M4            | Metal        | 34          | [ 293.97, 1857.80]   | 0.07             | max  | 5.83e-16 (2.57e-01,-)            | 6.70e-01 (5.23e-01,v) | 5.67e-01 (8.68e-01,-) |
| M3            | Metal        | 33          | [ 292.57, 1857.66]   | 0.10             | min  | 7.08e-16 (3.82e-01,-)            | 6.72e-01 (7.12e-01,-) | 5.69e-01 (1.29e+00,-) |
| M3            | Metal        | 33          | [ 292.57, 1857.66]   | 0.10             | max  | 6.97e-16 (3.82e-01,-)            | 6.72e-01 (7.12e-01,-) | 5.69e-01 (1.29e+00,-) |
| VIA3          | Via          | 53          | [ 293.97, 1857.73]   | 2                | min  | 1.04e-17 (5.50e-01,-)            | 6.71e-01 ( -, -)      | 5.68e-01 ( -, -)      |
| VIA3          | Via          | 53          | [ 293.97, 1857.73]   | 2                | max  | 4.37e-16 (5.50e-01,-)            | 6.71e-01 ( -, -)      | 5.68e-01 ( -, -)      |
| VIA4          | Via          | 54          | [ 293.97, 1857.87]   | 2                | min  | 5.83e-16 (5.50e-01,-)            | 6.70e-01 ( -, -)      | 5.67e-01 ( -, -)      |
| VIA4          | Via          | 54          | [ 293.97, 1857.87]   | 2                | max  | 3.33e-16 (5.50e-01,-)            | 6.70e-01 ( -, -)      | 5.67e-01 ( -, -)      |


```

The report shows the layer name, metal layer or via ID number, location coordinates, metal width or via cut number, analysis mode (min or max), electrical current value, and electromigration constraint value. The Mean, RMS, and Peak columns show the limit on the current imposed by the electromigration constraints and the actual current calculated for the wire segment or via at that location. The letter "v" indicates a violation and a hyphen indicates no violation for that location. A hyphen appearing in the Limit position means that there is no electromigration constraint for that metal or via layer.

To determine how the electromigration results were calculated for a particular net, use the `report_signal_em_calculation` command. For example,

```
icc_shell> report_signal_em_calculation n278
```

The report is similar to the report shown by the `report_signal_em -verbose` command, but the report applies only to the single specified net.

Identifying Unfixable Violations

If a signal electromigration violation cannot be fixed due to a user-specified setting, IC Compiler categorizes it as an unfixable violation. An unfixable violation can be caused by the violation being on a net that is frozen, a routing layer that is frozen, or a net shape that is frozen. If your design has unfixable violations, the `report_signal_em` command lists the total number of nets with unfixable violations. It also lists the number of nets in each category (frozen net, frozen layer, or frozen shape), as shown in the following example:

```
Warning: '2' nets with EM violation are unfixable because
      'they are frozen by NDR'.  (SI-145)
Warning: '7' nets with EM violation are unfixable because
      'the violations happen on frozen layers'.  (SI-145)
Warning: '3' nets with EM violation are unfixable because
      'the violations happen on frozen shapes'.  (SI-145)
```

If you use the `report_signal_em -violated -verbose` command, the generated report includes additional information for each unfixable violation, as shown in the following example:

```
Warning: Signal EM violation on net 'n51' is unfixable because it is
'layer frozen'. (SI-146)
```

A net can be frozen as a result of a routing rule specified by using the `set_net_routing_rule` command with the `-reroute frozen` option. To confirm this, use the `report_net_routing_rules` command.

```
icc_shell> report_net_routing_rules net_name -freeze
```

You can change the routing rule by reapplying the `set_net_routing_rule` command.

```
icc_shell> set_net_routing_rule net_name -reroute normal
```

A routing layer can be frozen as a result of routing options specified by using the `set_route_zrt_common_options` command with the `-freeze_layer_by_layer_name`, `-freeze_via_to_frozen_layer_by_layer_name`, and `-forbid_new_metal_by_layer_name` options. To confirm which layers and vias are frozen, use the `get_route_zrt_common_options` command.

```
icc_shell> get_route_zrt_common_options -name freeze_layer_by_layer_name
icc_shell> get_route_zrt_common_options \
    -name freeze_via_to_frozen_layer_by_layer_name
icc_shell> get_route_zrt_common_options \
    -name forbid_new_metal_by_layer_name
```

You can change the routing options by reapplying the `set_route_zrt_common_options` command.

```
icc_shell> set_route_zrt_common_options \
    -freeze_layer_by_layer_name {{layer_name false}} \
    -freeze_via_to_frozen_layer_by_layer_name {{layer_name false}} \
    -forbid_new_metal_by_layer_name {{layer_name false}}
```

A net shape can be frozen as a result of its `route_type` attribute being set to `User Enter`. To confirm this, use the `get_attribute` command.

```
icc_shell> get_attribute -class shape \
    [get_net_shapes -of [get_nets net_name]] route_type
```

You can change the `route_type` attribute by using the `set_attribute` command.

```
icc_shell> set_attribute -class shape \
    [get_net_shapes -of [get_nets net_name]] route_type "Signal Route"
```

Electromigration Fixing

The `fix_signal_em` command performs electromigration analysis and fixes any violations by modifying the problem nets using segment sizing, nondefault routing rules, or driver cell resizing. This command makes it unnecessary to manually run a repair file or to manually perform incremental routing to fix violations.

Before running the `fix_signal_em` command, make sure that the design is detail routed, the signal electromigration constraints are defined in the Milkyway design library, the switching activity is defined for all boundary nets, crosstalk analysis is enabled, and there are no timing or DRC violations. You can optionally use `report_signal_em` first to report electromigration violations before attempting to fix them.

This is the `fix_signal_em` command syntax:

```
fix_signal_em
  [-max_number_iterations count]
  [-only_segment_size]
  [-only_net_ndr]
  [-only_cell_based]
  [list_of_nets]
```

The command invokes electromigration analysis for nets, which calculates the current on every edge of the net and compares it with the current thresholds defined in the design library, and then attempts to fix any violations found while observing the DRC rules. The command options let you specify the maximum number of fixing iterations attempted and the types of fixing strategy applied: segment sizing, nondefault routing rules, or driver cell resizing.

To use the IC Compiler GUI to perform electromigration fixing, choose Route > Signal Electromigration > Fix Signal EM. The dialog box offers the same options as the `fix_signal_em` command.

9

Physical Datapath With Relative Placement

The IC Compiler physical datapath with relative placement capability provides a way for you to create structures in which you specify the relative column and row positions of instances. During placement and legalization, these structures, which are placement constraints called relative placement structures, are preserved and the cells in each structure are placed as a single entity. Relative placement is also called physical datapath and structured placement.

Note:

Relative placement is available in the IC Compiler package and IC Compiler-PC package, but not in the IC Compiler-XP package. Design Compiler topographical mode also supports relative placement.

The concepts and tasks necessary for doing relative placement within IC Compiler are described in these sections:

- [Introduction to Physical Datapath With Relative Placement](#)
- [Benefits of Relative Placement](#)
- [Flow for Relative Placement](#)
- [Considerations for Using Relative Placement](#)
- [Creating Relative Placement Groups](#)
- [Adding Objects to a Group](#)
- [Placement and Optimization of Relative Placement Groups](#)
- [Postplacement for Relative Placement Groups](#)

- Postplacement Optimization of Relative Placement Groups
- Analyzing the Relative Placement Results
- Working With Relative Placement Groups in the GUI
- Querying Relative Placement Groups and Objects
- Saving Relative Placement Information
- Ignoring Relative Placement Constraints
- Removing Relative Placement Groups
- Changing Relative Placement Information
- Deriving Relative Placement Groups
- Summary of Relative Placement Commands
- Limitations of Relative Placement

Introduction to Physical Datapath With Relative Placement

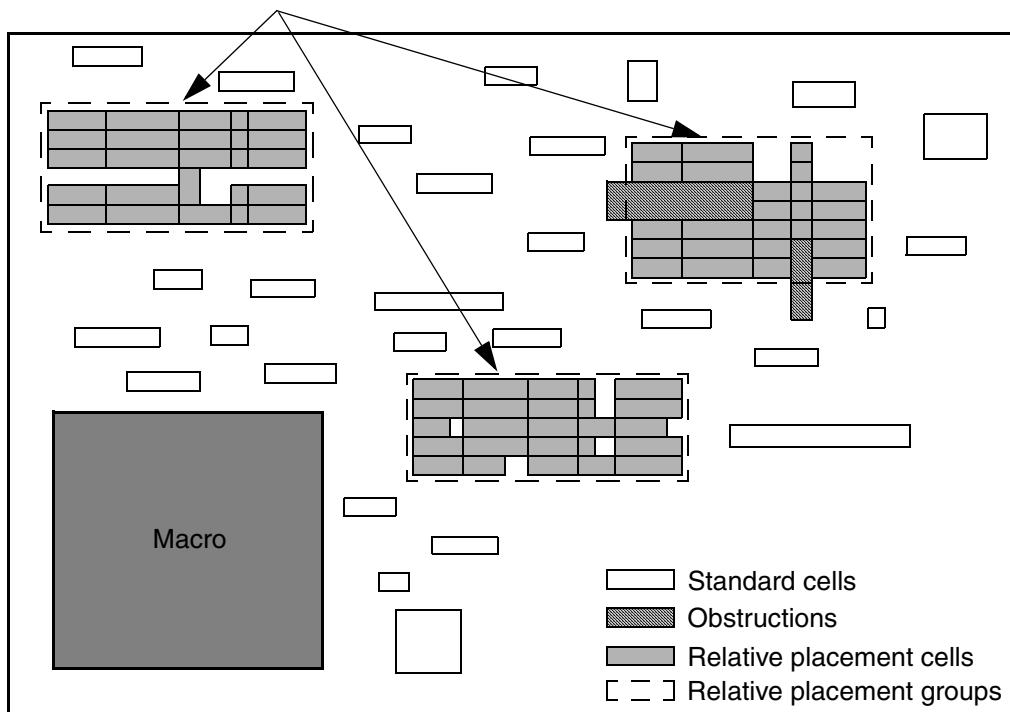
You perform relative placement by using a set of dedicated commands that specify the relative placement constraints, apply them to a gate-level netlist, check the constraints, remove the constraints, and write the annotated relative placement constraints to a script.

Relative placement is usually applied to datapaths and registers, but you can apply it to any cells in your design, controlling the exact relative placement topology of gate-level logic groups and defining the circuit layout. You can use relative placement to explore QoR benefits, such as shorter wire lengths, reduced congestion, better timing, skew control, fewer vias, better yield, and lower dynamic and leakage power.

The relative placement constraints that you create and annotate implicitly generate a matrix structure of the instances and control the placement of the instances. You use the resulting annotated netlist for physical optimization, during which IC Compiler preserves the structure and places it as a single entity or group, as shown in [Figure 9-1](#).

Figure 9-1 Relative Placement in a Floorplan

Relative placement groups
can be floating or fixed.



Benefits of Relative Placement

Along with being technology-independent and having the ability to improve routability, relative placement provides the following benefits:

- Reduces the placement search space in critical areas of the design, meaning greater predictability of QoR (wire length, timing, power, area) and congestion.
- Maintains relative placement during optimization, placement and legalization, clock tree synthesis, and routing.
- Provides a method for maintaining structured placement for legacy or intellectual property (IP) designs.
- Provides easy-to-use GUI support for datapath creation, analysis, and modification.
- Handles flat and hierarchical designs.

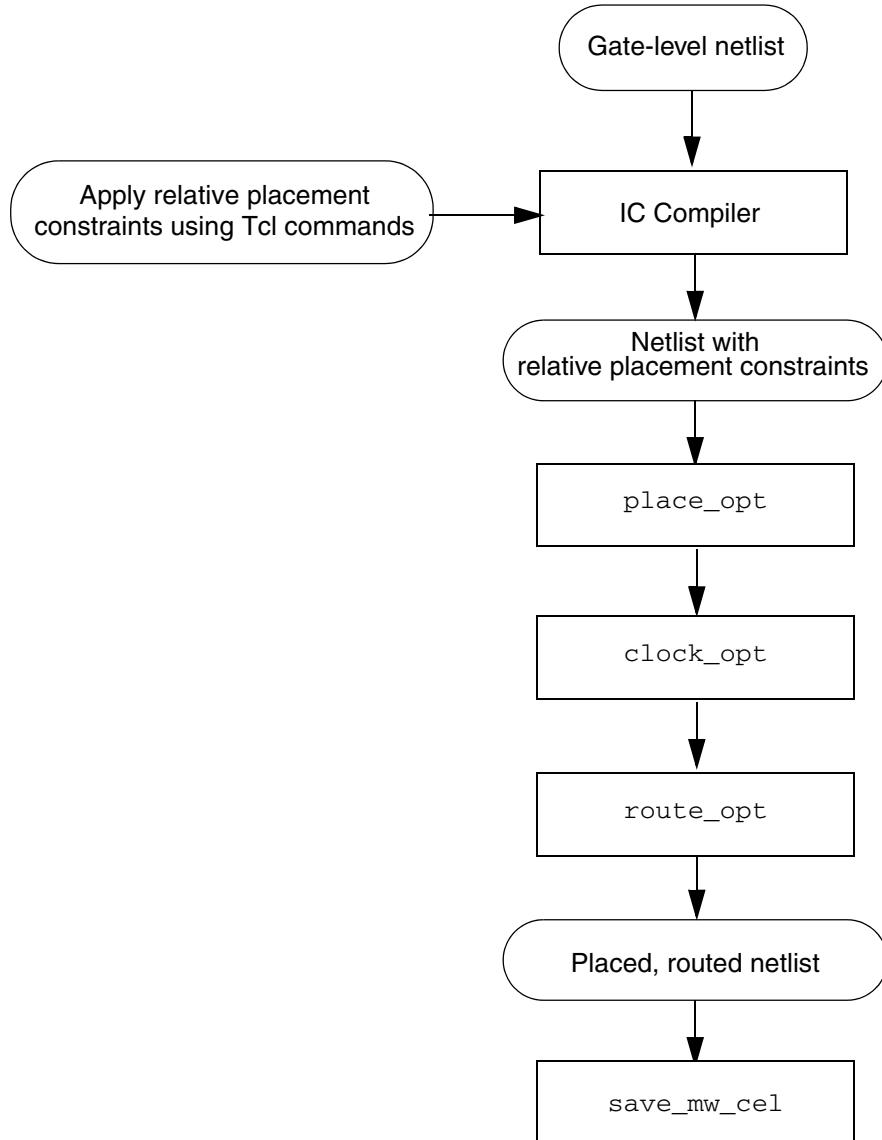
For complex designs, a typical design can have many engineers working on it and many blocks. Hierarchical relative placement makes it possible to place those blocks together relative to each other more easily. Any number of hierarchical levels is allowed.

- Allows sizing of relative placement cells while maintaining relative placement.

Flow for Relative Placement

A typical relative placement flow is shown in [Figure 9-2](#).

Figure 9-2 Relative Placement Flow



The flow for using relative placement follows these major steps:

1. Read a gate-level netlist into IC Compiler.
2. Specify and apply the gate-level relative placement constraints.
3. Generate the placed netlist.
4. Perform clock tree synthesis with the relative placement structures fixed in place.
5. Perform routing with the relative placement structures fixed in place.

Methodology for the Relative Placement Flow

The methodology for the relative placement flow follows these major steps:

1. Read the gate-level netlist into IC Compiler by using the `open_mw_cel` command.
2. Define the relative placement constraints.
 - o Create the relative placement groups by using the `create_rp_group` command.
See “[Creating Relative Placement Groups](#)” on page 9-10.
 - o Add relative placement objects to the groups by using the `add_to_rp_group` command.
See “[Adding Objects to a Group](#)” on page 9-21.

IC Compiler annotates the netlist with the relative placement constraints.

3. Prevent relative placement cells from being removed during optimization by using `psynopt_option` option. For example, enter

```
icc_shell> set_rp_group_options \
           -psynopt_option size_only \
           [get_rp_groups *]
```

See “[Preserving Relative Placement Structures](#)” on page 9-62.

4. Read the floorplan information. For example, enter

```
icc_shell> read_def top.def
```

5. Perform placement for the design by using the `place_opt` command.

Note:

Before you run the `place_opt` command, you can use the `create_placement` command to validate the relative placement results. After you achieve the desired relative placement results, run `place_opt` to perform placement and optimization.

6. Analyze the relative placement results.

You can analyze the relative placement results with either icc_shell or the IC Compiler GUI. See “[Analyzing the Relative Placement Results](#)” on page 9-65.

If the relative placement is not what you want, modify the relative placement constraints and run this procedure again.

7. Perform clock tree synthesis and physical optimization with the relative placement structures fixed in place. For example, enter

```
icc_shell> set_rp_group_options -cts_option fixed_placement \
           [get_rp_groups *]
icc_shell> clock_opt
```

See “[Postplacement Optimization of Relative Placement Groups](#)” on page 9-62.

8. Perform routing with the relative placement structures fixed in place. For example, enter

```
icc_shell> set_rp_group_options -route_opt_option fixed_placement \
           [get_rp_groups *]
icc_shell> route_opt
```

See “[Postplacement Optimization of Relative Placement Groups](#)” on page 9-62.

Script for a Relative Placement Flow

[Example 9-1](#) shows a script for running a relative placement flow.

Example 9-1 Script for the Relative Placement Flow

```
# Set library and design paths
source setup.tcl
open_mw_cel design_name

# Create relative placement constraints
create_rp_group ...
...
add_to_rp_group ...
...

# Apply design constraints
source constraints.tcl
# Read floorplan information
read_def top.def
# Perform placement, routing, and optimization on the design
place_opt

# Preserve relative placement during clock tree synthesis
set_rp_group_options -cts_option fixed_placement \
           [get_rp_groups *]
```

```
# Perform clock tree synthesis, routing, and optimization on the design
clock_opt

# Preserve relative placement during routing
set_rp_group_options -route_opt_option fixed_placement \
    [get_rp_groups *]

# Perform routing and optimization on the design
route_opt
```

The example uses

- The `set_rp_group_options -cts_option fixed_placement` command to preserve relative placement during clock tree synthesis
To allow cell sizing, you can specify the `size_only` keyword for the `-cts_option` option.
- The `set_rp_group_options -route_opt_option fixed_placement` command to preserve relative placement during routing
To allow cell sizing only when there is enough space, you can specify the `in_place_size_only` keyword for the `-route_opt_option` option.

Relative Placement Flow in Design Compiler Topographical Mode

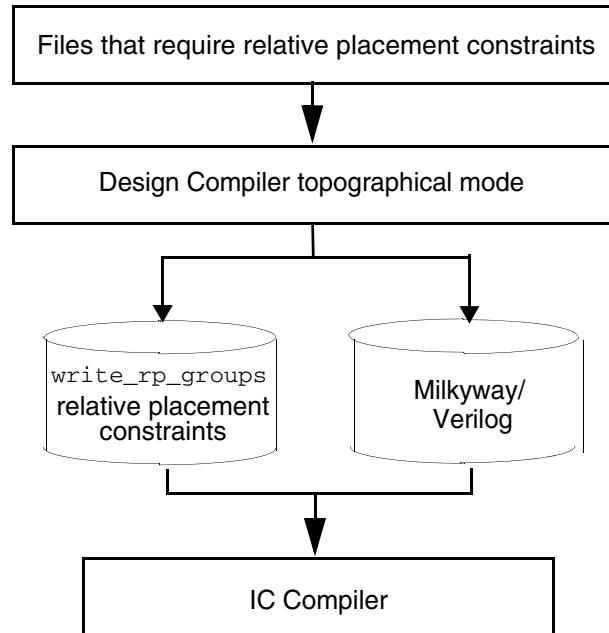
You can perform relative placement in Design Compiler topographical mode. You specify relative placement constraints by using a dedicated set of Tcl commands that are similar to the IC Compiler commands. In addition, you can place relative placement constraints in an RTL design, a GTECH netlist, or a mapped netlist by using the HDL compiler directives. Topographical mode annotates the netlist with the relative placement constraints. To save these constraints for subsequent IC Compiler sessions, use the `write_rp_groups` command.

If you import the design into the IC Compiler tool by using the `import_designs` command, use the `-rp_constraint` option to import the relative placement constraints. You can also import the relative placement constraints into the IC Compiler tool by sourcing the constraints file after you import the design.

For more information about using the Design Compiler tool to perform relative placement, see the *Design Compiler User Guide*.

[Figure 9-3](#) shows the relative placement flow in Design Compiler topographical mode.

Figure 9-3 Relative Placement Flow in Design Compiler Topographical Mode



Considerations for Using Relative Placement

When you use relative placement, keep the following points in mind:

- Relative placement requires a gate-level netlist. The format can be any format that IC Compiler can read.
- A design can contain both structured and unstructured objects (leaf cells, keepouts, hierarchical groups), and you can control which cells are to be structured by including the cells you want to structure in a relative placement group.

You must determine which portions of the design require a relative placement structure. Overly restricting the placement can produce poor results.

- Some designs, such as datapaths, are appropriate for structured placement, whereas others are more appropriate for usual placement. You must ensure that relative placement is applicable to your design.
- Relative placement is supported in multivoltage designs. In such designs, a relative placement group cannot cross voltage areas. For more information about multivoltage designs, see [Chapter 12, “Multivoltage Design Flow.”](#)

Creating Relative Placement Groups

A relative placement group is an association of cells, other groups, and keepouts. A group is defined by the number of rows and columns it uses.

Use the `create_rp_group` command to create a relative placement group. Use the `-columns` and `-rows` options to specify the number of columns and rows for the relative placement group. For example, to create a relative placement group that has six columns and six rows, use the following command:

```
icc_shell> create_rp_group rp1 -design designA -columns 6 -rows 6
```

The name of the relative placement group that the tool creates in the previous example is `designA::rp1`. The tool derives this name by combining the design name you provide by using the `-design` option, `designA` and the command argument you provide, `rp1`. You must use this name when referring to this group in subsequent relative placement commands.

[Figure 9-4](#) shows the positions of columns and rows in a relative placement group.

Figure 9-4 Relative Placement Column and Row Positions

row 5	0 5	1 5	2 5	3 5	4 5	5 5
row 4	0 4	1 4	2 4	3 4	4 4	5 4
row 3		1 3	2 3	3 3	4 3	5 3
row 2	0 2	1 2	2 2	3 2	4 2	5 2
row 1	0 1	1 1	2 1	3 1		5 1
row 0	0 0	1 0	2 0	3 0	4 0	5 0
	col 0	col 1	col 2	col 3	col 4	col 5

For the relative placement group in the previous figure,

- The column count begins from column 0 (the leftmost column).
- The row count begins from row 0 (the bottom row).
- The width of a column is the width of the widest cell in that column.
- The height of a row is the height of the tallest cell in that row.
- All positions in the structure are not used. For example, positions 0 3 (column 0, row 3) and 4 1 (column 4, row 1) are not used.

In addition to the number of columns and rows, you can also specify other properties of the relative placement group. [Table 9-1](#) describes the properties that you can specify and the corresponding option of the `create_rp_group` command that you should use.

Table 9-1 Specifying Relative Placement Group Properties

To do this	Use this option
Specify the type of alignment used by the group. You can specify either <code>bottom-left</code> (the default), <code>bottom-right</code> , or <code>bottom-pin</code> . See “ Aligning Leaf Cells Within a Column ” on page 9-23.	<code>-alignment</code>
Allow keepouts over tap cells. See “ Adding Keepouts ” on page 9-49.	<code>-allow_keepout_over_tapcell</code>
Allow nonrelative placement cells in relative placement groups. See “ Exposing Unused Space in Relative Placement Groups ” on page 9-62.	<code>-allow_non_rp_cells</code>
Specify a corner for the anchor point set by the <code>-x_offset</code> and <code>-y_offset</code> options. You can specify either <code>bottom-left</code> (the default), <code>bottom-right</code> , <code>top-left</code> , <code>top-right</code> , or <code>rp-location</code> . See “ Specifying a Corner for an Anchored Relative Placement Group ” on page 9-15.	<code>-anchor_corner</code>
Prevent inserting buffers inside the areas of placed relative placement groups. See “ Buffering Strategy for Relative Placement Groups ” on page 9-64.	<code>-auto_blockage</code>
Specify if the tool can automatically orient the cells in the specified relative placement group. See “ Orientation of Relative Placement Groups ” on page 9-30.	<code>-cell_orient_opt</code>
Specify the clock tree synthesis preservation attribute. You can specify <code>fixed_placement</code> or <code>size_only</code> . See “ Postplacement Optimization of Relative Placement Groups ” on page 9-62.	<code>-cts_option</code>

Table 9-1 Specifying Relative Placement Group Properties (Continued)

To do this	Use this option
Prevent inserting buffers into nets inside relative placement groups. See “Buffering Strategy for Relative Placement Groups” on page 9-64.	-disable_buffering
Specify the orientation of relative placement groups. You can specify default, N, FN, S, or FS. See “Orientation of Relative Placement Groups” on page 9-30.	-group_orient
Ignore this relative placement group. See “Ignoring Relative Placement Constraints” on page 9-91.	-ignore
Ignore site rows when legalizing the relative placement group. See “Ignoring Site Rows When Legalizing Relative Placement Groups” on page 9-59.	-ignore_rows
Specify the maximum allowable height in microns for the relative placement group. See “Constraining the Size of a Relative Placement Group for Optimization” on page 9-51.	-max_rp_height
Specify the maximum allowable width in microns for the relative placement group. See “Constraining the Size of a Relative Placement Group for Optimization” on page 9-51.	-max_rp_width
Legalize relative placement groups. You can specify low, medium, or high. See “Controlling Movement When Legalizing Relative Placement Groups” on page 9-59.	-move_effort
Specify the group alignment pin. See “Aligning Leaf Cells Within a Column” on page 9-23.	-pin_align_name
Specify how to treat fixed cells in the floorplan during legalization. See “Handling Fixed Cells During Relative Placement” on page 9-53.	-place_around_fixed_cells

Table 9-1 Specifying Relative Placement Group Properties (Continued)

To do this	Use this option
Specify a placement type for relative placement groups. You can specify <code>bit_slice</code> , <code>vertical_compression</code> , or <code>compression</code> . See “ Applying Compression to Relative Placement Groups ” on page 9-18.	<code>-placement_type</code>
Specify the physical optimization preservation attribute. You can specify <code>fixed_placement</code> , <code>size_only</code> , or <code>all_optimization</code> . See “ Postplacement Optimization of Relative Placement Groups ” on page 9-62.	<code>-psynopt_option</code>
Specify the routing preservation attribute. You can specify <code>fixed_placement</code> or <code>in_place_size_only</code> . See “ Postplacement Optimization of Relative Placement Groups ” on page 9-62.	<code>-route_opt_option</code>
Specify the utilization percentage (default is 100 percent).	<code>-utilization</code>
Specify the anchor location. See “ Anchoring Relative Placement Groups ” on page 9-13.	 <code>-x_offset</code> <code>-y_offset</code>

Alternatively, you can use the `set_rp_group_options` command to specify properties for a relative placement group. The `set_rp_group_options` command also supports all the options described in [Table 9-1](#).

After you create a relative placement group and specify the relevant properties, use the `add_to_rp_group` command to add objects (leaf cells, relative placement groups, or keepouts) to the group, as described in “[Adding Objects to a Group](#)” on page 9-21.

Anchoring Relative Placement Groups

By default, IC Compiler can place a relative placement group anywhere within the core area. You can control the placement of a top-level relative placement group by anchoring it.

To anchor a relative placement group, use the `create_rp_group` or the `set_rp_group_options` command with the `-x_offset` and `-y_offset` options. The offset values are float values, in microns, relative to the lower-left corner in the core area.

If you specify both the x- and y-coordinates, the group is anchored at that location. If you specify only one coordinate, IC Compiler can determine the placement by maintaining the

specified coordinate and sliding the group along the line passing through the unspecified coordinate.

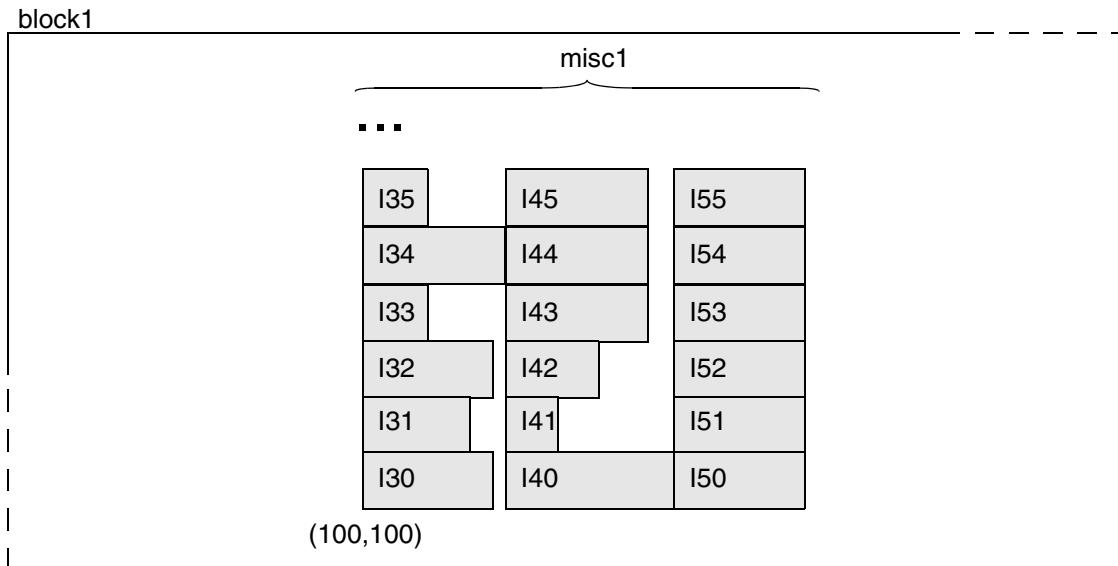
Note:

If you try to anchor a group that is not a top-level group or that causes an invalid placement, IC Compiler issues a warning message and continues relative placement. If you specify an anchor point that is outside the design boundary (or an anchor point that causes cells in the relative placement group to be outside the design boundary), IC Compiler issues a warning message and clusters the cells inside the design boundary.

For example, to specify a relative placement group anchored at (100, 100), as shown [Figure 9-5](#), enter the following command:

```
icc_shell> create_rp_group misc1 -design block1 \
    -columns 3 -rows 10 -x_offset 100 -y_offset 100
```

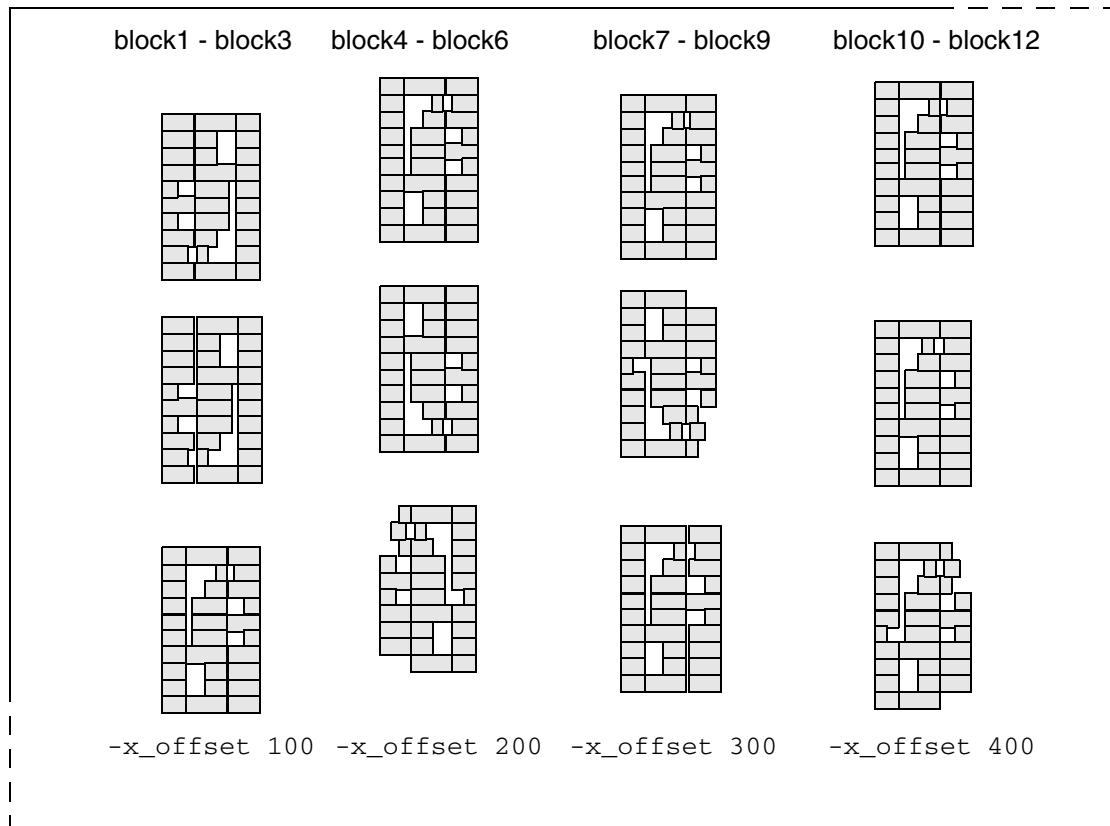
Figure 9-5 Anchored Relative Placement Group



The script in [Example 9-2](#) defines the locations of 12 relative placement groups that are aligned and anchored vertically at four coordinates, as shown in [Figure 9-6](#). For brevity, not every group is listed in the example.

Example 9-2 Definitions for Locations of Vertically Aligned and Anchored Groups

```
create_rp_group block1 -design misc1 -columns 3 -rows 10 -x_offset 100
create_rp_group block2 -design misc1 -columns 3 -rows 10 -x_offset 100
create_rp_group block3 -design misc1 -columns 3 -rows 10 -x_offset 100
create_rp_group block4 -design misc1 -columns 3 -rows 10 -x_offset 200
create_rp_group block5 -design misc1 -columns 3 -rows 10 -x_offset 200
create_rp_group block6 -design misc1 -columns 3 -rows 10 -x_offset 200
...
create_rp_group block12 -design misc1 -columns 3 -rows 10 -x_offset 400
```

Figure 9-6 Relative Placement Groups Aligned and Anchored Vertically

Specifying a Corner for an Anchored Relative Placement Group

To avoid obstructions, such as blockages, fixed cells, and tap arrays, you can set a corner for an anchored relative placement group. To specify the bottom-left, bottom-right, top-left or top-right corner for the anchor point set by the `-x_offset` and `-y_offset` options, use the `create_rp_group` or `set_rp_group_options` command with the `-anchor_corner` option.

You can specify

- The `-anchor_corner bottom-left` option

The anchor point of the relative placement group is set to the bottom-left corner. The default is the bottom-left corner.

- The `-anchor_corner bottom-right` option

The anchor point of the relative placement group is set to the bottom-right corner.

- The `-anchor_corner top-left` option

The anchor point of the relative placement group is set to the top-left corner.

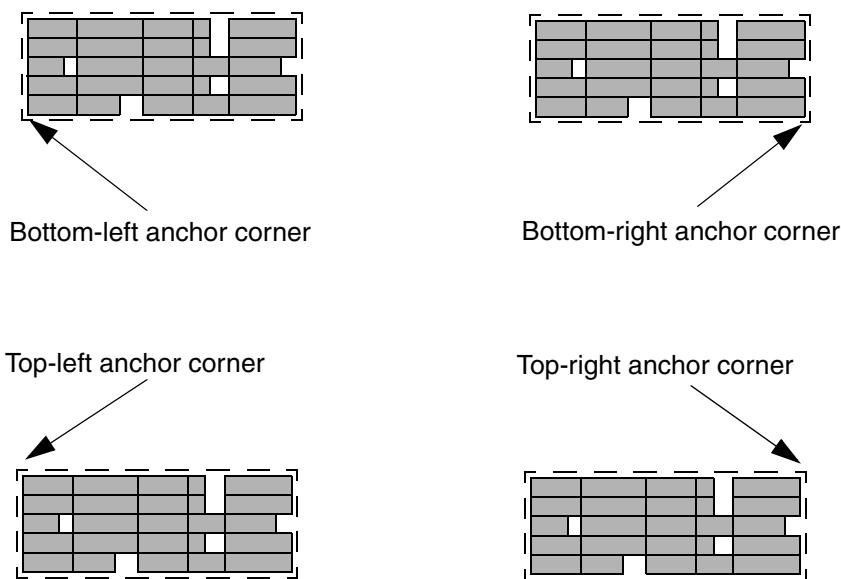
- The `-anchor_corner top-right` option

The anchor point of the relative placement group is set to the top-right corner.

For example, the following command specifies the bottom-right corner for the (100, 100) anchor point of the TOP::RP1 relative placement group.

```
icc_shell> set_rp_group_options TOP::RP1 \
           -anchor_corner bottom-right -x_offset 100 -y_offset 100
```

Figure 9-7 Bottom-Left, Bottom-Right, Top-Left and Top-Right Anchor Corners



In addition to the four corners of the relative placement group, you can also specify the bottom-left corner of any object within the relative placement group as the anchor point by using the `-anchor_corner rp-location` option along with the `-anchor_column` and `-anchor_row` options of the `create_rp_group` or `set_rp_group_options` commands.

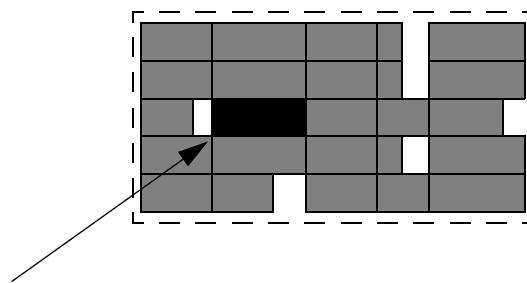
Note:

When you use the `-anchor_corner rp-location` option, the location specified by using the `-x_offset` and `-y_offset` options should not be empty. It should contain a cell, keepout, or a relative placement hierarchy.

For example, the following command specifies that the bottom-left corner of the cell at column 2, row 3 of the TOP::RP1 relative placement group should be anchored at location (100, 100).

```
icc_shell> set_rp_group_options TOP::RP3 \
    -anchor_corner rp-location -anchor_column 2 -anchor_row 3 \
    -x_offset 100 -y_offset 100
```

Figure 9-8 Selecting a Cell Within the Relative Placement Group to Anchor It



Selecting the cell at column 2, row 3 as the anchor corner for the relative placement group

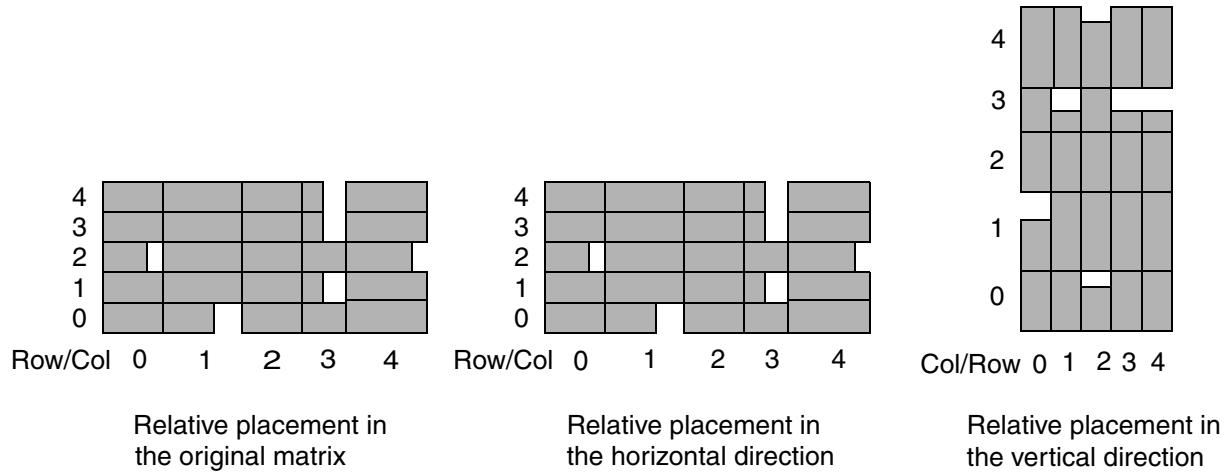
If you specify a value for the `-anchor_column` or `-anchor_row` option that is out of range for the corresponding relative placement group, the tool fails with an error message as shown in the following example:

```
Error: 'anchor_column' at column 6 is out of range for RP group TOP::RP3
that has only 5 columns. (RPGP-141)
```

Aligning Relative Placement Rows With the Site Row

By default, the IC Compiler tool aligns relative placement rows with the site rows of the design. Site rows can be either horizontal or vertical. If designs contain horizontal site rows, the tool places relative placement rows in the horizontal direction, following the specified anchor corner and alignment methods. For designs with vertical site rows, the tool places relative placement cells in a row lining up vertically such that the row is vertical and the column is horizontal for a relative placement group. Note that the `-alignment bottom-right` option of the `create_rp_group` and `set_rp_group_options` commands does not support vertical site rows.

[Figure 9-9](#) shows a relative placement group in the original matrix that is aligned in the horizontal direction and in the vertical direction.

Figure 9-9 Aligning Relative Placement in the Horizontal Direction and in the Vertical Direction

Applying Compression to Relative Placement Groups

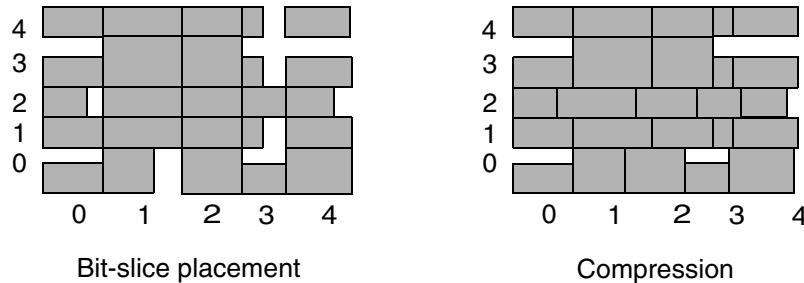
By default, the IC Compiler tool places relative placement groups using bit-slice placement. In addition to bit-slice placement, you can apply compression to relative placement groups in the vertical direction or in the horizontal direction. To change the placement method or apply compression, set the `-placement_type` option to one of the following keywords with the `create_rp_group` or `set_rp_group_options` command:

- `bit-slice`

Setting the `-placement_type` option to `bit_slice` enables the placer to place the next row after the tallest object in the row and the next column after the widest object in the column. Both row alignment and column alignment are preserved. The default is `bit_slice`.

- `compression`

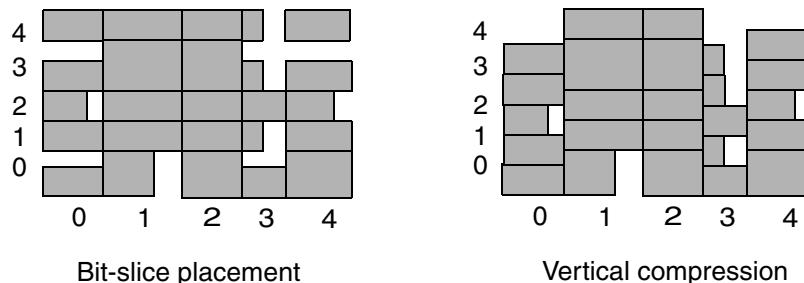
Setting the `-placement_type` option to `compression` enables bit-stack placement, which places each row in a relative placement group without any gaps between leaf cells, lower-level hierarchical relative placement groups, and keepouts, as shown in [Figure 9-10](#). Column alignment is not preserved.

Figure 9-10 Bit-Slice Placement Versus Compression

If you specify the `-utilization` option and set the `-placement_type` option to `compression`, the IC Compiler tool observes the utilization constraint but leaves gaps between leaf elements in a relative placement row. If you set the `-placement_type` option to `compression` and the `-alignment` option to `bottom-right` or `bottom_pin`, IC Compiler issues an error message. To disable compression, set the `-placement_type` option to an appropriate keyword, such as `bit_slice`, with the `set_rp_group_options` or `create_rp_group` command.

- `vertical_compression`

Setting the `-placement_type` option to `vertical_compression` enables vertical compression such that no gap is allowed between leaf cells, lower-level hierarchical relative placement groups, and keepouts in a column, as shown in [Figure 9-11](#). Applying vertical compression reduces the wire lengths of routes in a column. Row alignment is not preserved.

Figure 9-11 Bit-Slice Placement Versus Vertical Compression

For example, the following command sets vertical compression to the `d::rp` relative placement group:

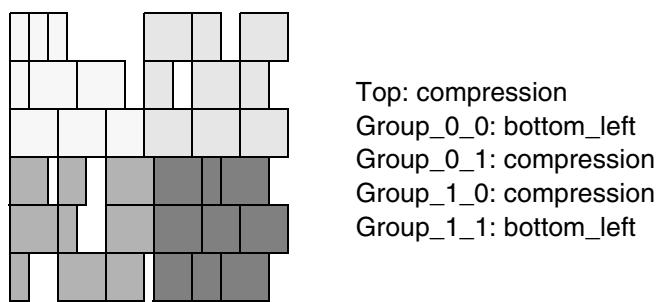
```
icc_shell> set_rp_group_options d::rp
          -placement_type vertical_compression
```

The setting of the `-placement_type` option does not propagate from a parent group to child groups.

Handling Relative Placement Groups With Mixed Alignment

You can place relative placement groups with alignment and relative placement groups with a compression constraint in the same top-level group, as shown in [Figure 9-12](#). The individual groups of the top level are aligned with compression only if you set the `-placement_type` option to `compression`. Note that the compression specified at the top level does not propagate to the child groups. Bottom-left alignment is the default for the top-level groups.

Figure 9-12 Compression of Relative Placement Groups With Mixed Alignment



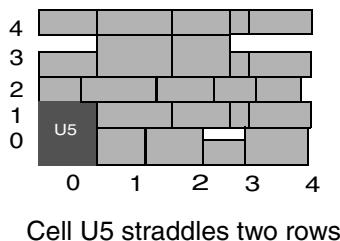
Handling Relative Placement Groups With Straddling Leaf Cells

You can apply compression to a relative placement group with cells that straddle multiple rows or columns, as shown in the following example:

```
icc_shell> add_to_rp_group d::rp -leaf U5 \
    -column 0 -row 0 -num_columns 1 -num_rows 2
icc_shell> set_rp_group_options d::rp -placement_type compression
```

[Figure 9-13](#) shows the placement of the relative placement group in the previous example.

Figure 9-13 Compression of a Relative Placement Group With a Cell That Straddles Multiple Rows



For information about adding cells that straddle multiple rows or columns to a relative placement group, see [“Adding Leaf Cells” on page 9-21](#).

Adding Objects to a Group

You can add leaf cells, other relative placement groups, and keepouts to relative placement groups that have been created with the `create_rp_group` command. You can add objects from the command line by using the `add_to_rp_group` command, in the relative placement hierarchy browser, or in the relative placement editing dialog box. For information about the relative placement hierarchy browser, see “[Using the Relative Placement Hierarchy Browser](#)” on page 9-69. For information about the relative placement editing dialog box, see “[Editing and Creating Relative Placement Groups](#)” on page 9-73.

When you add an object to a relative placement group, keep the following points in mind:

- The relative placement group to which you are adding the object must exist.
 - The object must be added to an empty location in the relative placement group.
-

Adding Leaf Cells

When you add leaf cells to a relative placement group, you specify their column position, row position, and orientation within the relative placement group. By default, relative placement respects the intercell spacing rules that are defined for standard cells.

To add leaf cells in the relative placement hierarchy browser (Placement > New RP Hierarchy View),

1. Select the group you want to add to in the left pane.
2. Select the Grid tab in the right pane.
3. Select the desired location in the grid view.

When you add leaf cells, make sure that the selected location is empty.

4. Right-click and choose Add Object.

The Add Objects to RP dialog box appears.

5. Select Add Leaf Cell, specify the leaf cell name, and select the possible orientations.
6. Click OK.

You can also add leaf cells by using the relative placement editing dialog box in the GUI. For more information, see “[Editing and Creating Relative Placement Groups](#)” on page 9-73.

Alternatively, you can add a leaf cell to a relative placement group by using the `add_to_rp_group` command.

In a relative placement group, a leaf cell can occupy multiple column positions or multiple row positions, which is known as leaf cell straddling. You can create a more compact relative

placement group by straddling leaf cells. To define straddling, you specify multiple column or row positions by using the `-num_columns` or `-num_rows` options respectively. You can specify a maximum of 255 rows and 255 columns. If you do not set these options, the default is 1 for both column and row positions. For example, to create a leaf cell of two columns and one row, enter

```
icc_shell> add_to_rp_group design1::rp1 -leaf U23 \
    -column 0 -num_columns 2 -row 0 -num_rows 1
```

You should not place a relative placement keepout at the same location of a straddling leaf cell. In addition, straddling is for leaf cells only, but not for hierarchical groups or keepouts.

Note:

You can apply right alignment or pin alignment to a straddling leaf cell with multiple row positions, but not to a cell with multiple column positions.

Specifying Orientations for Leaf Cells

You can specify orientations for leaf cells when you add them to a relative placement group. If you do not specify a leaf cell orientation, IC Compiler automatically assigns a legal orientation for the leaf cells.

To specify the orientation for leaf cells,

- Select the possible orientations in the Add Objects to RP dialog box.
or
- Include the `-orientation` option with a list of possible orientations when you add the cells to the group with the `add_to_rp_group` command.
or
- Set the `rp_orientation` attribute on leaf cells by using the `set_attribute` command.

The tool chooses one legal orientation from the list of orientations that you provide.

For more information about setting orientations for leaf cells, see “[Controlling Relative Placement Cell Orientations](#)” on page 9-34.

Aligning Leaf Cells Within a Column

You can align the leaf cells in a column of a relative placement group by using the following alignment methods:

- Bottom left (default)
- Bottom right
- Pin alignment
- Via region alignment

Controlling the cell alignment can improve the timing and routability of your design.

Aligning by Bottom-Left Corners

By default, the IC Compiler tool aligns the leaf cells by aligning the bottom-left corners. (To explicitly specify this alignment method, use the `-alignment bottom-left` option of the `create_rp_group` or `set_rp_group_options` command.)

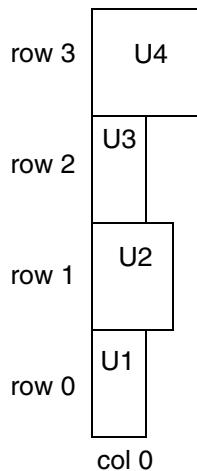
```
icc_shell> set_rp_group_options -alignment bottom-left [get_rp_groups *]
```

The script in [Example 9-3](#) defines a relative placement group that is bottom-left aligned. The resulting structure is shown in [Figure 9-14](#).

Example 9-3 Definition for Bottom-Left-Aligned Relative Placement Group

```
create_rp_group rp1 -design pair_design -columns 1 -rows 4
  add_to_rp_group pair_design::rp1 -leaf U1 -column 0 -row 0
  add_to_rp_group pair_design::rp1 -leaf U2 -column 0 -row 1
  add_to_rp_group pair_design::rp1 -leaf U3 -column 0 -row 2
  add_to_rp_group pair_design::rp1 -leaf U4 -column 0 -row 3
```

Figure 9-14 Bottom-Left-Aligned Relative Placement Group



Aligning by Bottom-Right Corners

To align a group by aligning the bottom-right corners, use the `-alignment bottom-right` option of the `create_rp_group` or `set_rp_group_options` command.

```
icc_shell> set_rp_group_options -alignment bottom-right \
[get_rp_groups *]
```

Note:

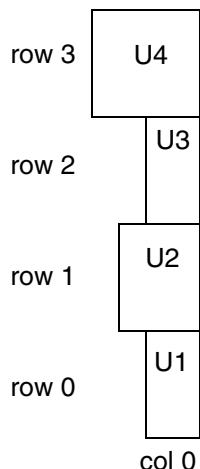
For hierarchical relative placement groups, the bottom-right alignment does not propagate through the hierarchy.

The script in [Example 9-4](#) defines a relative placement group that is bottom-right aligned. The resulting structure is shown in [Figure 9-15](#).

Example 9-4 Definition for Bottom-Right-Aligned Relative Placement Group

```
create_rp_group rp1 -design pair_design -columns 1 -rows 4 \
-alignment bottom-right
add_to_rp_group pair_design::rp1 -leaf U1 -column 0 -row 0
add_to_rp_group pair_design::rp1 -leaf U2 -column 0 -row 1
add_to_rp_group pair_design::rp1 -leaf U3 -column 0 -row 2
add_to_rp_group pair_design::rp1 -leaf U4 -column 0 -row 3
```

Figure 9-15 Bottom-Right-Aligned Relative Placement Group



Aligning by Pin Location

To align a group by pin location, use the `-alignment bottom-pin` and `-pin_align_name` options of the `create_rp_group` or `set_rp_group_options` command.

```
icc_shell> set_rp_group_options -alignment bottom-pin \
-pin_align_name align_pin
```

IC Compiler looks for the specified alignment pin in each cell in the column. If the alignment pin exists in a cell, the cell is aligned by use of the pin location. If the specified alignment pin

does not exist in a cell, the cell is aligned at the bottom-left corner, and IC Compiler issues an information message. If the specified alignment pin does not exist in any cell in the column, IC Compiler issues a warning message.

If you specify both pin alignment and cell orientation, IC Compiler resolves potential conflicts as follows:

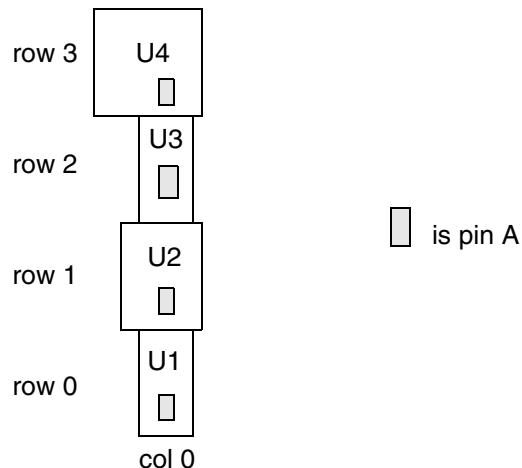
- User specifications for cell orientation take precedence over the pin alignment performed by IC Compiler.
- Pin alignment done by IC Compiler takes precedence over the cell-orientation optimization done by IC Compiler.

The script in [Example 9-5](#) defines a relative placement group that is aligned by pin A. The resulting structure is shown in [Figure 9-16](#).

Example 9-5 Definition for Relative Placement Group Aligned by Pins

```
create_rp_group rp1 -design pair_design -columns 1 -rows 4 \
    -pin_align_name A
add_to_rp_group pair_design::rp1 -leaf U1 -column 0 -row 0
add_to_rp_group pair_design::rp1 -leaf U2 -column 0 -row 1
add_to_rp_group pair_design::rp1 -leaf U3 -column 0 -row 2
add_to_rp_group pair_design::rp1 -leaf U4 -column 0 -row 3
```

Figure 9-16 Relative Placement Group Aligned by Pins



When you specify an alignment pin for a group, the pin applies to all cells in the group. You can override the group alignment pin for specific cells in the group by specifying the `-pin_align_name` option when you use the `add_to_rp_group` command to add the cells to the group.

Note:

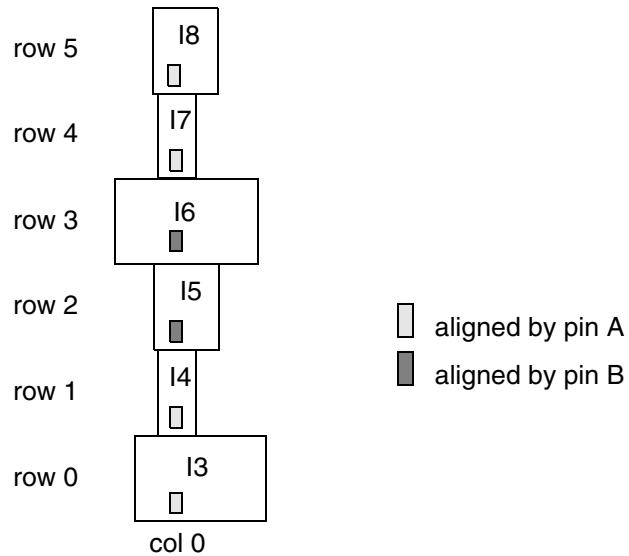
You cannot specify a cell-specific alignment pin when you add a leaf cell from the relative placement hierarchy browser.

The script in [Example 9-6](#) defines relative placement group misc1, which uses pin A as the group alignment pin; however, instances I5 and I6 use pin B as their alignment pin, rather than the group alignment pin. The resulting structure is shown in [Figure 9-17](#).

Example 9-6 Definition for Aligning a Group and Leaf Cells by Pins

```
create_rp_group misc1 -design block1 -columns 3 -rows 10 \
    -pin_align_name A
add_to_rp_group block1::misc1 -leaf I3 -column 0 -row 0
add_to_rp_group block1::misc1 -leaf I4 -column 0 -row 1
add_to_rp_group block1::misc1 -leaf I5 -column 0 -row 2 \
    -pin_align_name B
add_to_rp_group block1::misc1 -leaf I6 -column 0 -row 3 \
    -pin_align_name B
add_to_rp_group block1::misc1 -leaf I7 -column 0 -row 4
add_to_rp_group block1::misc1 -leaf I8 -column 0 -row 5
```

Figure 9-17 Relative Placement Group Aligned by Pins



Aligning by Via Regions

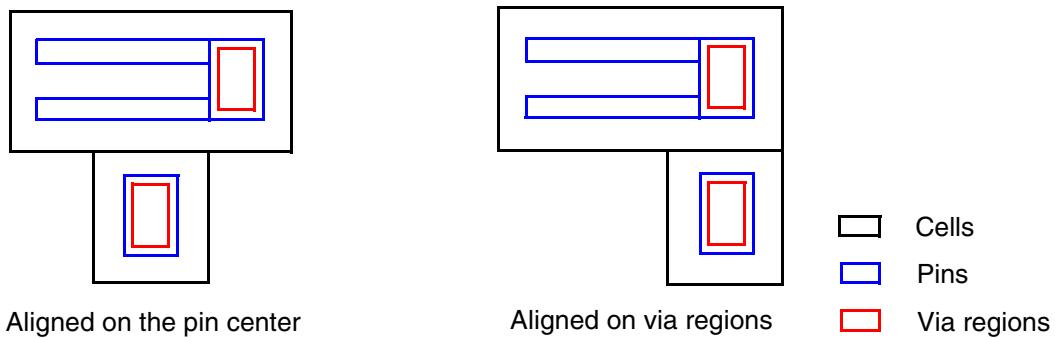
The IC Compiler tool aligns cells at the center of the specified alignment pins. When via regions are available for the specified alignment pins in a relative placement group, the tool automatically aligns cells based on the individual via regions of the pins rather than the center of the pins.

Alignment by via regions provides the following benefits:

- Makes straight routes easier for the router.

[Figure 9-18](#) shows the alignment of two cells, using methods that are based on the center of the specified pins and on the via regions of the specified pins.

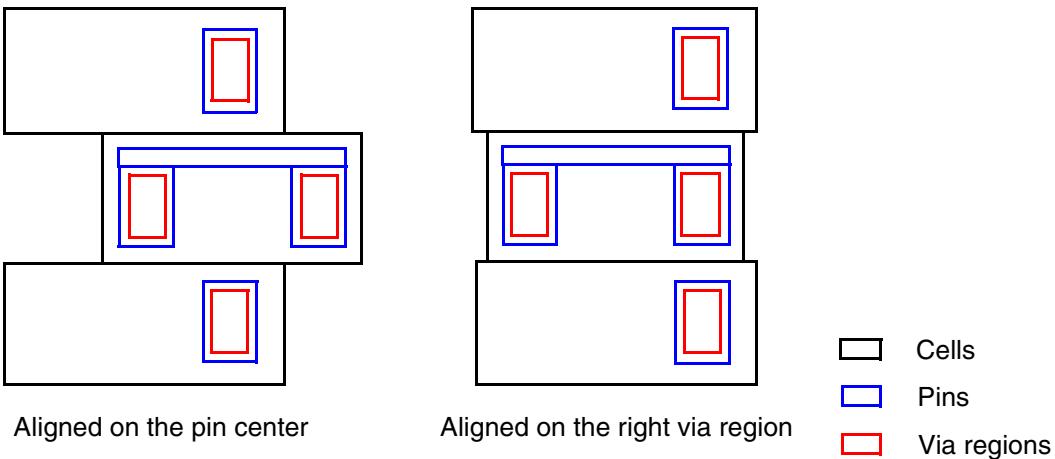
Figure 9-18 Alignment by Pins Versus Alignment by Via Regions



- Minimizes the column width of a relative placement group when a pin has more than one via region.

[Figure 9-19](#) shows the alignment of three cells where the middle cell has two via regions.

Figure 9-19 Alignment by Pins Versus Alignment by the Right Via Region



Adding a Free Placement Area for Unstructured Cells

You can mix structured and unstructured cells within a relative placement group by adding a free placement area for unstructured cells. Use the `add_to_rp_group` command with the `-free_placement` option to add a free placement area. During placement, the tool places the unstructured cells in the free placement area.

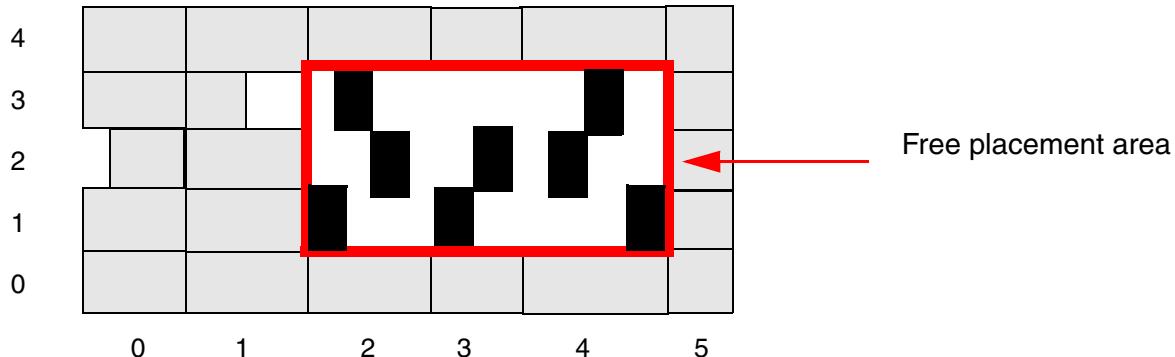
Use the `-num_columns` and `-num_rows` options to specify the width and height of the free placement area and the `-column` and `-row` options to specify the location of its lower-left corner.

The following example shows how to add a free placement area that is three columns wide and three rows high at location (2,1), to the relative placement group named `design1::rp1`:

```
icc_shell> add_to_rp_group design1::rp1 -leaf [get_cell hrm22*] \
    -column 2 -row 1 -num_columns 3 -num_rows 3 -free_placement
```

[Figure 9-20](#) shows the free placement area added to the relative placement group `design1::rp2` in the previous example.

Figure 9-20 Free Placement Area Specified Within the Relative Placement Group



When you use the `-num_columns` option, the tool calculates the actual width of the free placement area by adding the width of the corresponding columns. Similarly, when you use the `-num_rows` option, the tool calculates the actual height of the free placement area by adding the height of the corresponding rows.

For example, for the free placement area in [Figure 9-20](#), the actual width is the sum of the width of columns 2, 3, and 4. The actual height is the sum of the height of rows 1, 2, and 3.

You can specify the actual width and height of a free placement area by using the `-width` and `-height` options. The units for the `-width` and `-height` options are the width and height of a placement site.

If the tool is unable to calculate the actual width and height of the free placement area by using the `-num_columns` and `-num_rows` options, you must specify the width and height by using the `-width` and `-height` options.

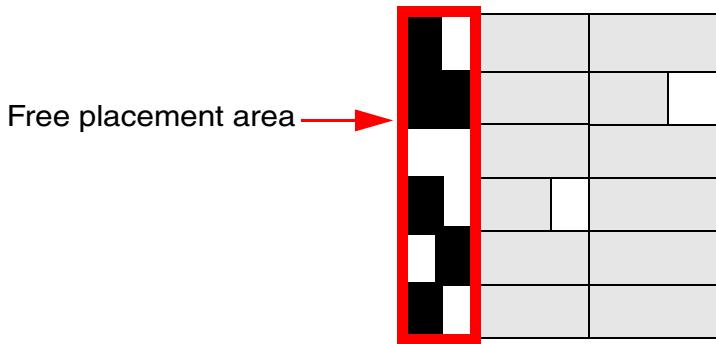
If you add a free placement area that occupies an entire column, as shown in [Figure 9-21](#), you must specify the width by using the `-width` option. If you specify the width by using the `-num_columns` option, the tool cannot calculate the actual width of the free placement area because there are no structured cells in that column.

The following example shows how to add a free placement area that is a boundary column, with a width equivalent to that of 12 placement sites:

```
icc_shell> add_to_rp_group design1::rp2 -leaf [get_cell mst5*] \
    -column 0 -row 0 -width 12 -num_rows 6 -free_placement
```

[Figure 9-21](#) shows the free placement area added to the relative placement group `design1::rp2` in the previous example.

Figure 9-21 Adding a Free Placement Area as a Boundary Column



If you add a free placement area that occupies an entire row, as shown in [Figure 9-22](#), you must specify the height by using the `-height` option. If you specify the height by using the `-num_rows` option, the tool cannot calculate the actual height of the free placement area because there are no structured cells in that row.

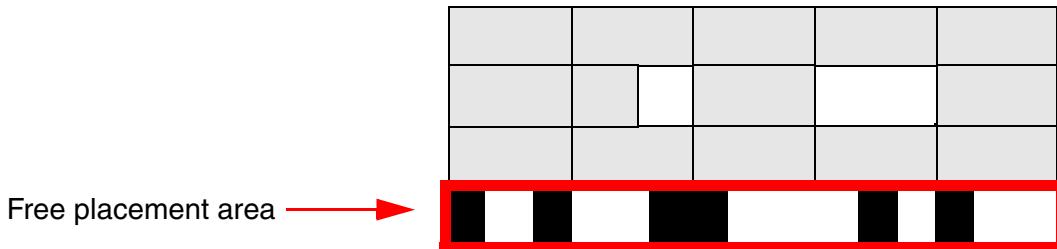
The following example shows how you add a free placement area that is a boundary row, with a height equivalent to that of 1 placement site:

```
icc_shell> add_to_rp_group design1::rp3 -leaf [get_cell BBL*] \
    -column 0 -row 0 -num_columns 5 -height 1 -free_placement
```

[Figure 9-22](#) shows the free placement area added to the relative placement group `design1::rp3` in the previous example.

If you specify both the `-width` and `-num_columns` options, the tool uses the `-width` option only if it cannot calculate the actual width by using the `-num_columns` option. Similarly, if you specify both the `-height` and `-num_rows` options, the tool uses the `-height` option only if it cannot calculate the actual height by using the `-num_rows` option.

Figure 9-22 Adding a Free Placement Area as a Boundary Row



After you add a free placement area to a relative placement group, use the `check_rp_groups` command to check for the following issues:

- The tool cannot determine the height or width of the free placement area based on the options you specified
- The area defined is too small to accommodate the specified unstructured cells

You can modify the size of an existing free placement area by using the `modify_rp_groups` command. To specify the free placement area you want to modify, use the `-free_placement` option along with the column and row location of its lower-left corner. To modify the height, use the `-height` option along with the desired height. To modify the width, use the `-width` option along with the desired width. The units for the `-height` and `-width` options are the height and width of a placement site.

The following example shows how to modify the height and width of the free placement area that is at location (2,1) of the relative placement group named `design1::rp1`:

```
icc_shell> modify_rp_groups [get_rp_groups design1::rp1] \
    -free_placement {2 1} -height 5 -width 4
```

Orientation of Relative Placement Groups

The orientation of relative placement groups is determined according to the data flow. If the data flow is from left to right, the first column is placed on the left side, and subsequent columns are placed toward the right. If the data flow is from right to left, the first column is placed on the right side, and subsequent columns are placed toward the left. Normally, row position is from bottom to top: that is, the first row is placed at the bottom, and subsequent rows are placed toward the top. If the row position of the data flow is from top to bottom, the first row is placed at the top, and subsequent rows are placed toward the bottom.

Relative Placement Group Orientations

The IC Compiler tool allows relative placement groups to be oriented in any of the following ways:

- North (N)

The column position of the relative placement group is from left to right, and the row position is from bottom to top. This orientation is considered as north (N).

- Flipped north (FN)

The column position of the relative placement group is from right to left, and the row position is from bottom to top; that is, the orientation of the group is flipped with respect to its north orientation. This orientation is considered as flipped north (FN).

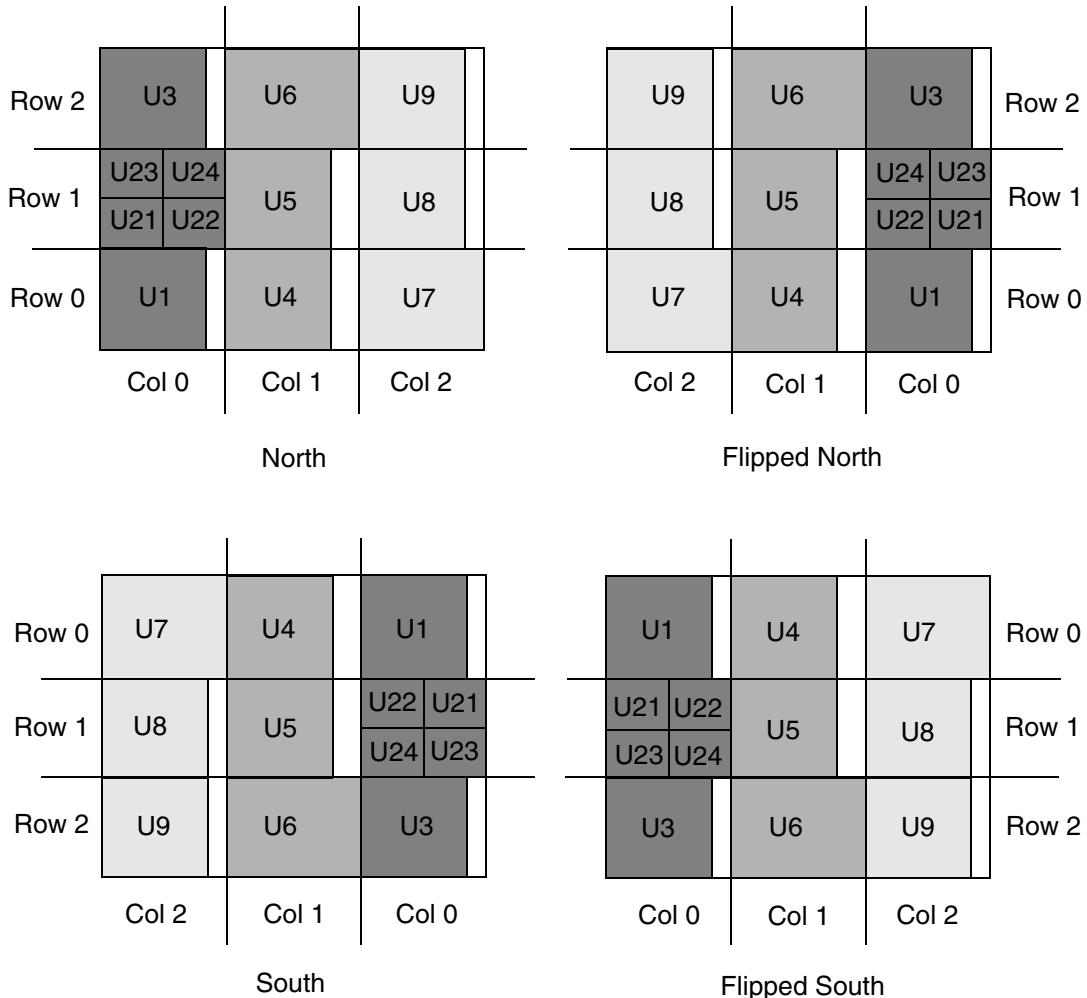
- South (S)

The column position of the relative placement group is from right to left, and the row position is from top to bottom. This orientation is considered as south (S).

- Flipped south (FS)

The column position of the relative placement group is from left to right, and the row position is from top to bottom; that is, the group is flipped with respect to its south orientation. This orientation is considered as flipped south (FS).

[Figure 9-23](#) shows how the column and row positions in a relative placement group are placed for the four orientations.

Figure 9-23 Orientation of Relative Placement Groups

The orientation of relative placement groups is automatically set by the tool to minimize wire length. You can also choose to set the orientation of relative placement groups by using the `set_rp_group_options` command.

For example, the following command sets the relative placement group orientation to north.

```
icc_shell> set_rp_group_options [get_rp_groups design::rp] \
-group_orient N
```

For designs with hierarchical relative placement groups, the orientation settings are propagated down to the lowest level in hierarchy.

Note:

When the orientation of a relative placement group is changed, the constraints on the relative placement group, such as alignment and utilization, are preserved according to the specifications that you provide.

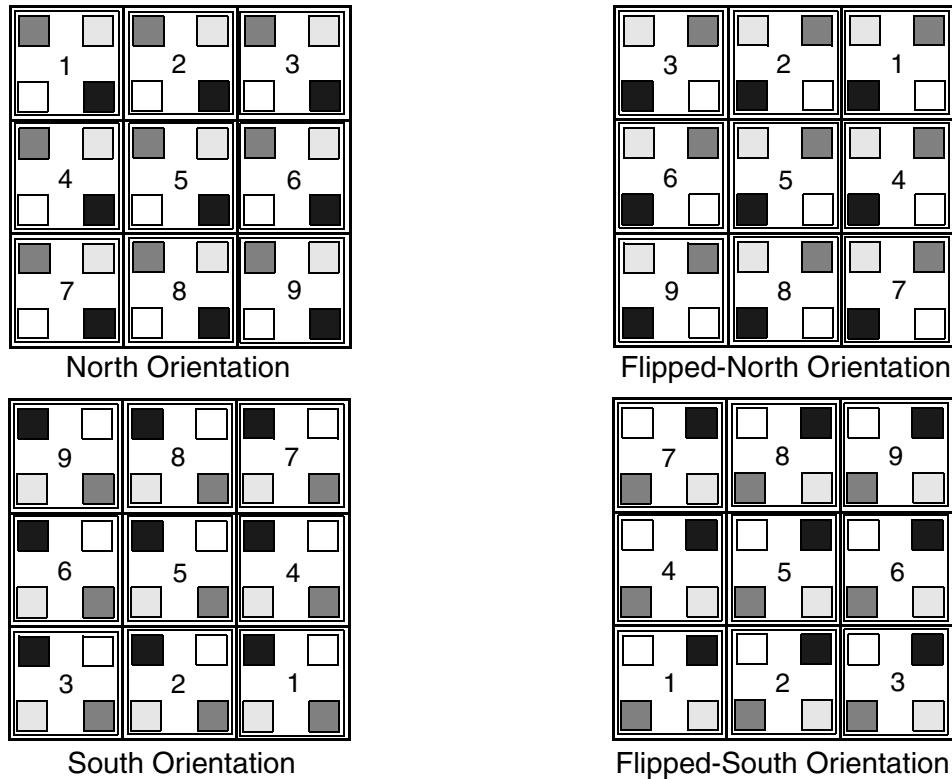
Relative Placement Cell Orientations

By default, the tool places relative placement cells according to the orientation of the relative placement group that contains the relative placement cells. When the orientation of the relative placement group is changed, the relative placement cell orientations are changed automatically to minimize wire length.

For example,

- When the relative placement group is flipped from a north (N) to flipped-north (FN) orientation or south (S) to flipped-south (FS) orientation, the relative placement cell orientations are also flipped accordingly.
- When the relative placement group is changed from a north (N) to south (S) orientation, the tool rotates the relative placement cells 180 degrees counterclockwise if the rotation is legal. If a relative placement cell was previously in its north orientation, it is in its south orientation after the change of orientation of the relative placement group.
- When the relative placement group is changed from a north (N) to flipped-south (FS) orientation, the relative placement cells are rotated 180 degrees counterclockwise if the rotation is legal and then flipped. If a relative placement cell was previously in its north orientation, it is in its flipped-south orientation after the change of orientation of the relative placement group.

[Figure 9-24](#) shows the orientations of relative placement cells when the orientation of the relative placement group that contains the relative placement cells is changed.

Figure 9-24 Orientations of Relative Placement Cells

Controlling Relative Placement Cell Orientations

By default, the tool orients relative placement cells according to the orientation of the relative placement group that contains the relative placement cells. You can enable the tool to place the cells automatically in the most appropriate orientation to optimize wire length. To enable this capability, you specify the `-cell_orient_opt` option of the `create_rp_group` or `set_rp_group_options` command. The `-cell_orient_opt` option is disabled by default.

To remove the setting of automatic cell orientation, use the `-cell_orient_opt` option of the `remove_rp_group_options` command.

You can explicitly set the orientations of relative placement cells by using the `set_attribute` command with the following syntax:

```
set_attribute rp_cell_name rp_orientation direction
```

For example, the following command allows the tool to choose N or FS orientation for the relative placement cell U14 in the relative placement group that has the north orientation.

```
icc_shell> set_attribute U14 rp_orientation {N FS}
```

For more information, see “[Changing the Attributes of Relative Placement Cells](#)” on page 9-100.

Adding Relative Placement Groups

Hierarchical relative placement allows relative placement groups to be embedded within other relative placement groups. The embedded groups then are handled similarly to leaf cells.

You can use hierarchical relative placement to simplify the expression of relative placement constraints. With hierarchical relative placement, you do not need to provide relative placement information multiple times for a recurring pattern.

There are two methods for adding a relative placement group to a hierarchical group:

- Include the group

If the relative placement group to be added is in the same design as its parent group, it is an included group. You can include groups in either flat or hierarchical designs. See “[Including Relative Placement Groups](#)” on page 9-36.

- Instantiate the group

If the relative placement group to be added is in an instance of a subdesign of its parent group, it is an instantiated group. You can instantiate groups only in hierarchical designs. See “[Instantiating Relative Placement Groups](#)” on page 9-38.

The following sections describe these aspects of hierarchical relative placement groups:

- [Benefits of Hierarchical Relative Placement Groups](#)
- [Including Relative Placement Groups](#)
- [Instantiating Relative Placement Groups](#)
- [Using Hierarchical Relative Placement for Straddling](#)
- [Using Hierarchical Relative Placement for Compression](#)
- [Effect of Ungrouping on Hierarchical Relative Placement](#)
- [Effect of Uniquifying on Hierarchical Relative Placement](#)

Benefits of Hierarchical Relative Placement Groups

Using hierarchical relative placement provides these benefits:

- Allows you to organize your relative placement in a manner that is easier to maintain and understand. For example, you can create the relative placement group to parallel your Verilog or VHDL organization.
- Allows reuse of a repeating placement pattern, such as an adder.
- Can reduce the number of lines of relative placement information you need to write.
- Allows integrating blocks.
- Provides flexibility for the configuration you want.

Including Relative Placement Groups

To include relative placement groups in the relative placement hierarchy browser (Placement > New RP Hierarchy View),

1. Select the group you want to add to in the left pane.
2. Select the Grid tab in the right pane.
3. Select the desired location in the grid view.

To include relative placement groups, make sure that the selected location is empty.

4. Right-click and choose Add Object.

The Add Objects to RP dialog box appears.

5. Specify the row and column position, specify the alignment method, select Add Hierarchical RP, and specify the relative placement group name.
6. Click OK.

Alternatively, you can use the `add_to_rp_group` command. The syntax for including a group in a hierarchical group by using `add_to_rp_group` is

```
add_to_rp_group hier_group -hierarchy group_name
    [-column col_number] [-row row_number]
    [-alignment bottom-left | bottom-right]
```

The group specified in the `-hierarchy` option must be in the same design as the hierarchical group in which you are including it.

When you include a relative placement group in a hierarchical group, it is as if the included group is directly embedded within its parent group. An included group can be used only in a group of the same design and only one time. However, a group that contains an included

group can be further included in another group in the same design or can be instantiated in a group of a different design.

You can also add relative placement groups by using the relative placement editing dialog box. For more information, see [“Editing and Creating Relative Placement Groups” on page 9-73](#).

The script in [Example 9-7](#) creates a hierarchical group (rp4) that contains three included groups (rp1, rp2, and rp3). Groups rp1, rp2, rp3, and rp4 are all in the design top. The contents of groups rp1, rp2, and rp3 are treated as leaf cells when they are included in group rp4. You can further include group rp4 in another group in the design top, or you can instantiate group rp4 in a group of a different design.

The construction of the resulting hierarchical relative placement group is shown in [Figure 9-25](#).

Example 9-7 Including Groups in a Hierarchical Group

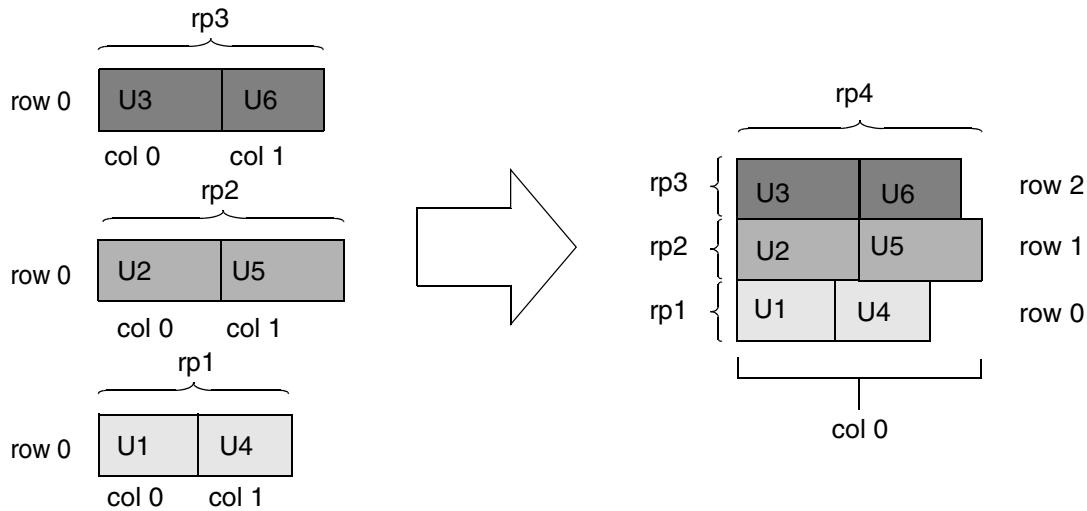
```
create_rp_group rp1 -design top -columns 2 -rows 1
    add_to_rp_group top::rp1 -leaf U1 -column 0 -row 0
    add_to_rp_group top::rp1 -leaf U4 -column 1 -row 0

create_rp_group rp2 -design top -columns 2 -rows 1
    add_to_rp_group top::rp2 -leaf U2 -column 0 -row 0
    add_to_rp_group top::rp2 -leaf U5 -column 1 -row 0

create_rp_group rp3 -design top -columns 2 -rows 1
    add_to_rp_group top::rp3 -leaf U3 -column 0 -row 0
    add_to_rp_group top::rp3 -leaf U6 -column 1 -row 0

create_rp_group rp4 -design top -columns 1 -rows 3
    add_to_rp_group top::rp4 -hierarchy top::rp1 \
        -column 0 -row 0
    add_to_rp_group top::rp4 -hierarchy top::rp2 \
        -column 0 -row 1
    add_to_rp_group top::rp4 -hierarchy top::rp3 \
        -column 0 -row 2
```

Figure 9-25 Including Groups in a Hierarchical Group



Instantiating Relative Placement Groups

The syntax for instantiating a group in a hierarchical group is

```
add_to_rp_group hier_group
  -hierarchy group_name -instance instance_name
  -orientation direction
  [-column col_number] [-row row_number] ]
```

Note:

You cannot instantiate relative placement groups from the relative placement hierarchy browser; you must do this from the command line.

The group specified in the *-hierarchy* option must be defined in the reference design of the instance specified in the *-instance* option. In addition, the specified instance must be in the same design as the hierarchical group in which you are instantiating the specified group.

Note:

If you define a relative placement group in a subdesign that has been instantiated multiple times, only those instances that are instantiated in a hierarchical relative placement group have relative placement. Instances that are not instantiated in a hierarchical relative placement group do not have relative placement.

Using an instantiated group is a useful way to replicate relative placement information across multiple instances of a design and to create relative placement relationships between

those instances. An instantiated group can be used multiple times and in multiple places. For example, use instantiated groups for these cases:

- You have multiple relative placement layouts you want to use for different instances of a design.
- You have only one layout but want to specify relative placement between instances of that layout or between instances and other cells and groups.

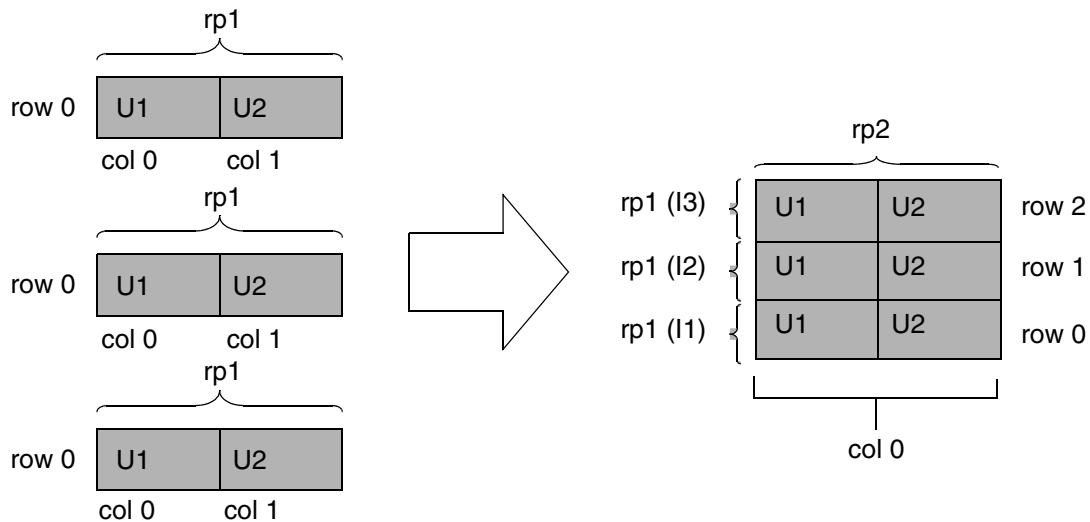
The script in [Example 9-8](#) creates a hierarchical group (rp2) that contains three instances of group rp1. Group rp1 is in the design pair_design and includes leaf cells U1 and U2. Group rp2 is a hierarchical group in the design mid_design that instantiates group rp1 three times (mid_design must contain at least three instances of pair_design). Group rp2 is treated as a leaf cell. You can instantiate rp2 multiple times and in multiple places, up to the number of times mid_design is instantiated in your netlist.

The construction of the resulting hierarchical relative placement group is shown in [Figure 9-26](#).

Example 9-8 Instantiating Groups in a Hierarchical Group

```
create_rp_group rp1 -design pair_design -columns 2 -rows 1
    add_to_rp_group pair_design::rp1 -leaf U1 -column 0 -row 0
    add_to_rp_group pair_design::rp1 -leaf U2 -column 1 -row 0

create_rp_group rp2 -design mid_design -columns 1 -rows 3
    add_to_rp_group mid_design::rp2 \
        -hierarchy pair_design::rp1 -instance I1 -column 0 -row 0
    add_to_rp_group mid_design::rp2 \
        -hierarchy pair_design::rp1 -instance I2 -column 0 -row 1
    add_to_rp_group mid_design::rp2 \
        -hierarchy pair_design::rp1 -instance I3 -column 0 -row 2
```

Figure 9-26 Instantiating Groups in a Hierarchical Group

Setting the Orientation of an Instantiated Relative Placement Group

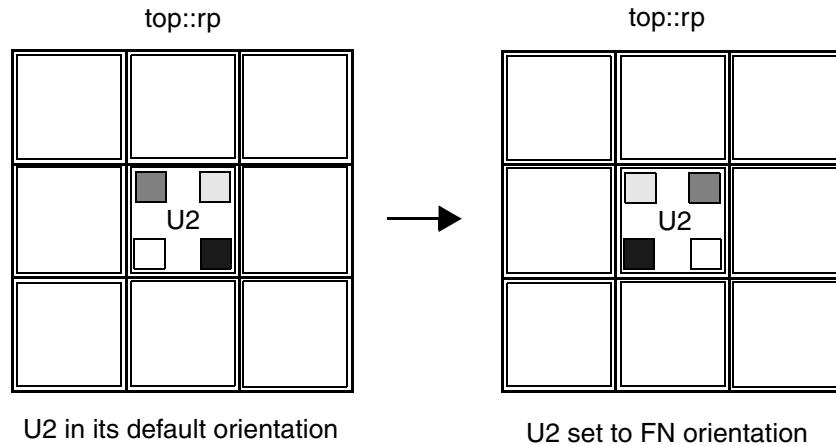
To specify the orientation of an instantiated relative placement group in a hierarchical relative placement group, set the `-orientation`, `-hierarchy`, and `-instance` options with the `add_to_rp_group` command. You can set the orientation to one of the following four ways: north (N), flipped north (FN), south (S), and flipped south (FS). If you do not specify the orientation, the instantiated relative placement group assumes its orientation from the definition by default. You cannot use the `-orientation` option on included groups.

For example, the following command adds the `grp::ripple` relative placement group that is instantiated with the `U2` name and set for the flipped-north orientation to the `top::rp` hierarchical relative placement group:

```
icc_shell> add_to_rp_group top::rp \
    -hierarchy grp::ripple -instance U2 -orientation FN
```

[Figure 9-27](#) shows the orientation of the `U2` relative placement group before and after the command performs the flipped-north operation.

Figure 9-27 Example of Setting the Orientation of an Instantiated Relative Placement Group



Changing the Structures of Relative Placement Groups

To modify the structures of existing relative placement groups, use the `modify_rp_groups` command to add, remove, swap, and flip rows and columns.

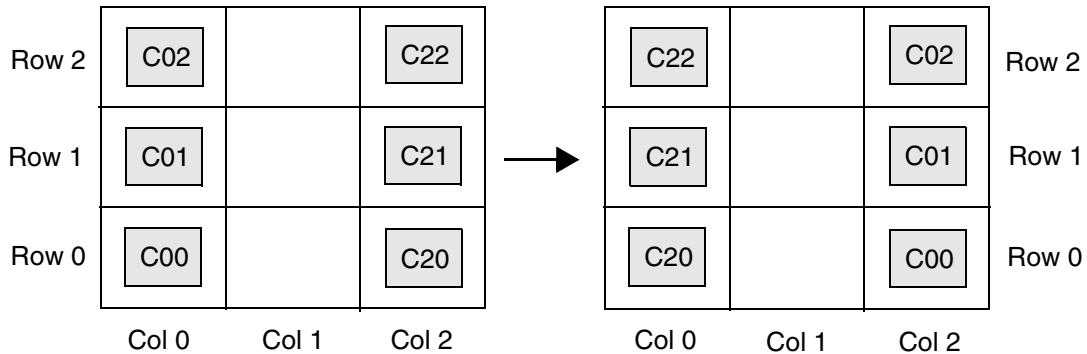
The syntax is

```
modify_rp_groups rp_groups
  [-add_rows_at integer]
  [-add_columns_at integer]
  [-remove_rows_at integer]
  [-remove_columns_at integer]
  [-number integer]
  [-flip_row integer]
  [-flip_column integer]
  [-swap_rows list]
  [-swap_columns list]
```

To add a row or column to a relative placement group, use the `-add_rows_at` or `-add_columns_at` option respectively. To remove a row or column, use the `-remove_rows_at` or `-remove_columns_at` option respectively. To flip a row or column, use the `-flip_row` or `-flip_column` option respectively. To specify multiple rows or columns to be removed or added, set the `-number` option to a value greater than one. The default is one. To swap two rows or columns, use the `-swap_rows` or `-swap_columns` option respectively.

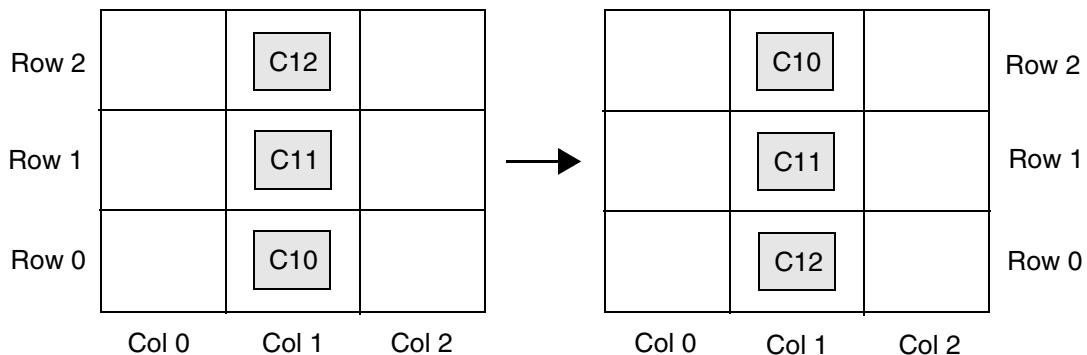
For example, to swap the first and third columns of the `design::my_rp_group` relative placement group, as shown in [Figure 9-28](#), enter

```
icc_shell> modify_rp_groups [get_rp_groups design::my_rp_group] \
  -swap_columns {0 2}
```

Figure 9-28 Swapping Columns of a Relative Placement Group

To flip the second column of the design::my_rp_group relative placement group, as shown in [Figure 9-28](#), enter

```
icc_shell> modify_rp_groups [get_rp_groups design::my_rp_group] \
    -flip_column 1
```

Figure 9-29 Flipping a Column of a Relative Placement Group

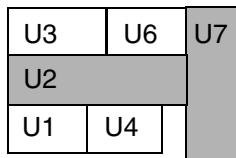
Alternatively, you can perform these operations by using the relative placement view window in the GUI. For more information, see [“Editing the Structures of Relative Placement Groups” on page 9-81](#).

Using Hierarchical Relative Placement for Straddling

A cell can occupy multiple column positions or multiple row positions, which is known as straddling. To define cells for straddling, use the method for including a relative placement group, which is described in “[Including Relative Placement Groups](#) on page 9-36”. For more information about leaf cell straddling, see “[Adding Leaf Cells](#)” on page 9-21.

[Figure 9-30](#) shows a relative placement group in which cells straddle columns (instance U2) and rows (instance U7).

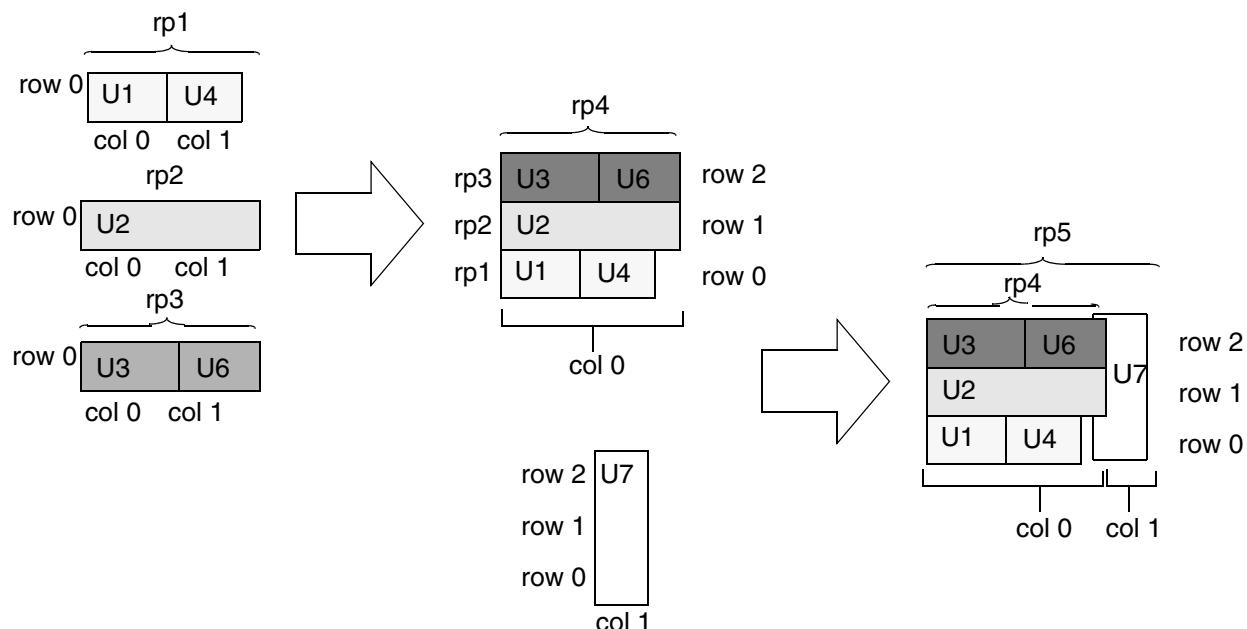
Figure 9-30 Hierarchical Relative Placement Group With Straddling



[Figure 9-31](#) shows the process of using hierarchical relative placement to build this structure. First, define relative placement groups that contain the leaf cells: rp1 contains U1 and U4, rp2 contains U2, and rp3 contains U3 and U6. Then define a group (rp4) that contains these groups. Finally, define a group that contains the hierarchical group rp4 and the leaf cell U7. The resulting group includes both the column and the row straddle.

[Example 9-9](#) shows the commands used in this process.

Figure 9-31 Straddling With Hierarchical Relative Placement



Example 9-9 Straddling With Hierarchical Relative Placement

```

create_rp_group rp1 -design top -columns 2 -rows 1
    add_to_rp_group top::rp1 -leaf U1 -column 0 -row 0
    add_to_rp_group top::rp1 -leaf U4 -column 1 -row 0

create_rp_group rp2 -design top -columns 1 -rows 1
    add_to_rp_group top::rp2 -leaf U2 -column 0 -row 0

create_rp_group rp3 -design top -columns 2 -rows 1
    add_to_rp_group top::rp3 -leaf U3 -column 0 -row 0
    add_to_rp_group top::rp3 -leaf U6 -column 1 -row 0

create_rp_group rp4 -design top -columns 1 -rows 3
    add_to_rp_group top::rp4 -hierarchy top::rp1 -column 0 -row 0
    add_to_rp_group top::rp4 -hierarchy top::rp2 -column 0 -row 1
    add_to_rp_group top::rp4 -hierarchy top::rp3 -column 0 -row 2

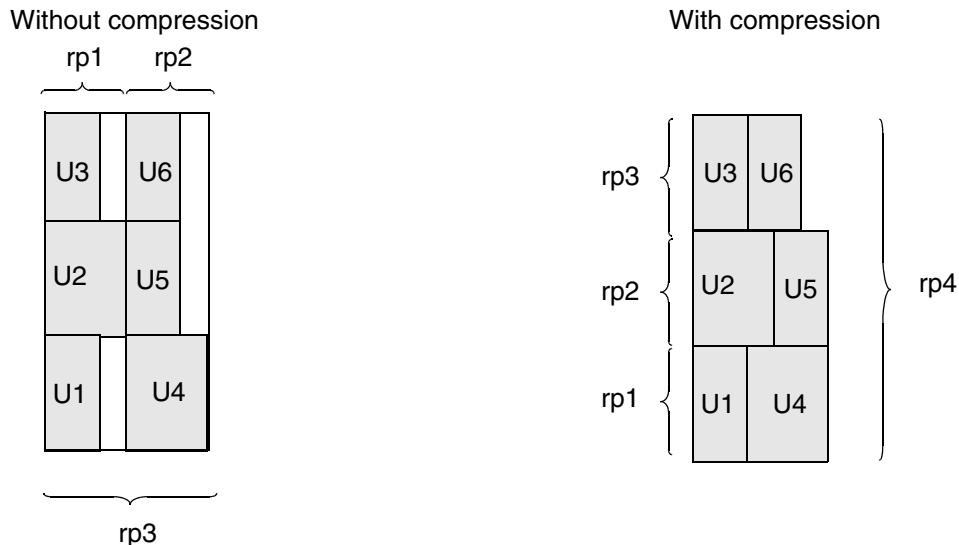
create_rp_group rp5 -design top -columns 2 -rows 1
    add_to_rp_group top::rp5 -hierarchy top::rp4 -column 0 -row 0
    add_to_rp_group top::rp5 -leaf U7 -column 1 -row 0

```

Using Hierarchical Relative Placement for Compression

By default, construction for relative placement aligns cells from their bottom-left corner. Compression removes empty space in rows to create a more compact structure. The compressed columns are no longer aligned, and utilization is higher in the area of the compressed cells. To define a compressed structure, use the method for including a relative placement group (see “[Including Relative Placement Groups](#)” on page 9-36).

[Figure 9-32](#) shows the same cells aligned without compression and with compression. To define the compressed structure, first define relative placement groups that contain the rows of leaf cells: rp1 contains U1 and U4, rp2 contains U2 and U5, and rp3 contains U3 and U6. Then define a group (rp4) that contains these groups. [Example 9-10](#) shows the commands used to build the compressed structure.

Figure 9-32 Bottom-Left Alignment Construction and Compression**Example 9-10 Compression With Hierarchical Relative Placement**

```

create_rp_group rp1 -design top -columns 2 -rows 1
    add_to_rp_group top::rp1 -leaf U1 -column 0 -row 0
    add_to_rp_group top::rp1 -leaf U4 -column 1 -row 0

create_rp_group rp2 -design top -columns 2 -rows 1
    add_to_rp_group top::rp2 -leaf U2 -column 0 -row 0
    add_to_rp_group top::rp2 -leaf U5 -column 1 -row 0

create_rp_group rp3 -design top -columns 2 -rows 1
    add_to_rp_group top::rp3 -leaf U3 -column 0 -row 0
    add_to_rp_group top::rp3 -leaf U6 -column 1 -row 0

create_rp_group rp4 -design top -columns 1 -rows 3
    add_to_rp_group top::rp4 -hierarchy top::rp1 -column 0 -row 0
    add_to_rp_group top::rp4 -hierarchy top::rp2 -column 0 -row 1
    add_to_rp_group top::rp4 -hierarchy top::rp3 -column 0 -row 2

```

Alternatively, you can apply compression in the horizontal direction by setting the `-placement_type` option to `compression` with the `create_rp_group` or `set_rp_group_options` command. See “[Applying Compression to Relative Placement Groups](#)” on page 9-18.

Effect of Ungrouping on Hierarchical Relative Placement

The `ungroup` command changes the hierarchical relative placement structure. When you ungroup the current design, use `ungroup -flatten -all`, to ensure that the relative placement gets flattened properly.

After you ungroup, instantiated relative placement groups are converted to included relative placement groups, because the design is flattened and all the groups are now of the same design. Instantiation of hierarchical modules no longer exists.

Relative placement groups affected by an `ungroup` command are renamed to show the path to the group before flattening, followed by a slash (/) and the original group name. If this results in a name collision, a numbered suffix is added to create a unique name.

For example, [Figure 9-33](#) shows a hierarchical relative placement group that contains multiple instantiations of group rp1 before and after ungrouping. Before ungrouping, the relative placement definition is as shown in [Example 9-11](#). After ungrouping, the relative placement definition is as shown in [Example 9-12](#).

Example 9-11 Instantiated Groups Before Ungrouping

```
create_rp_group rp1 -design pair_design -columns 2 -rows 1
  add_to_rp_group pair_design::rp1 -leaf U1 -column 0 -row 0
  add_to_rp_group pair_design::rp1 -leaf U2 -column 1 -row 0

create_rp_group rp2 -design mid_design -columns 1 -rows 3
  add_to_rp_group mid_design::rp2 -hierarchy pair_design::rp1 \
    -instance I1 -column 0 -row 0
  add_to_rp_group mid_design::rp2 -hierarchy pair_design::rp1 \
    -instance I2 -column 0 -row 1
  add_to_rp_group mid_design::rp2 -hierarchy pair_design::rp1 \
    -instance I3 -column 0 -row 2
```

Example 9-12 Instantiated Groups After Ungrouping

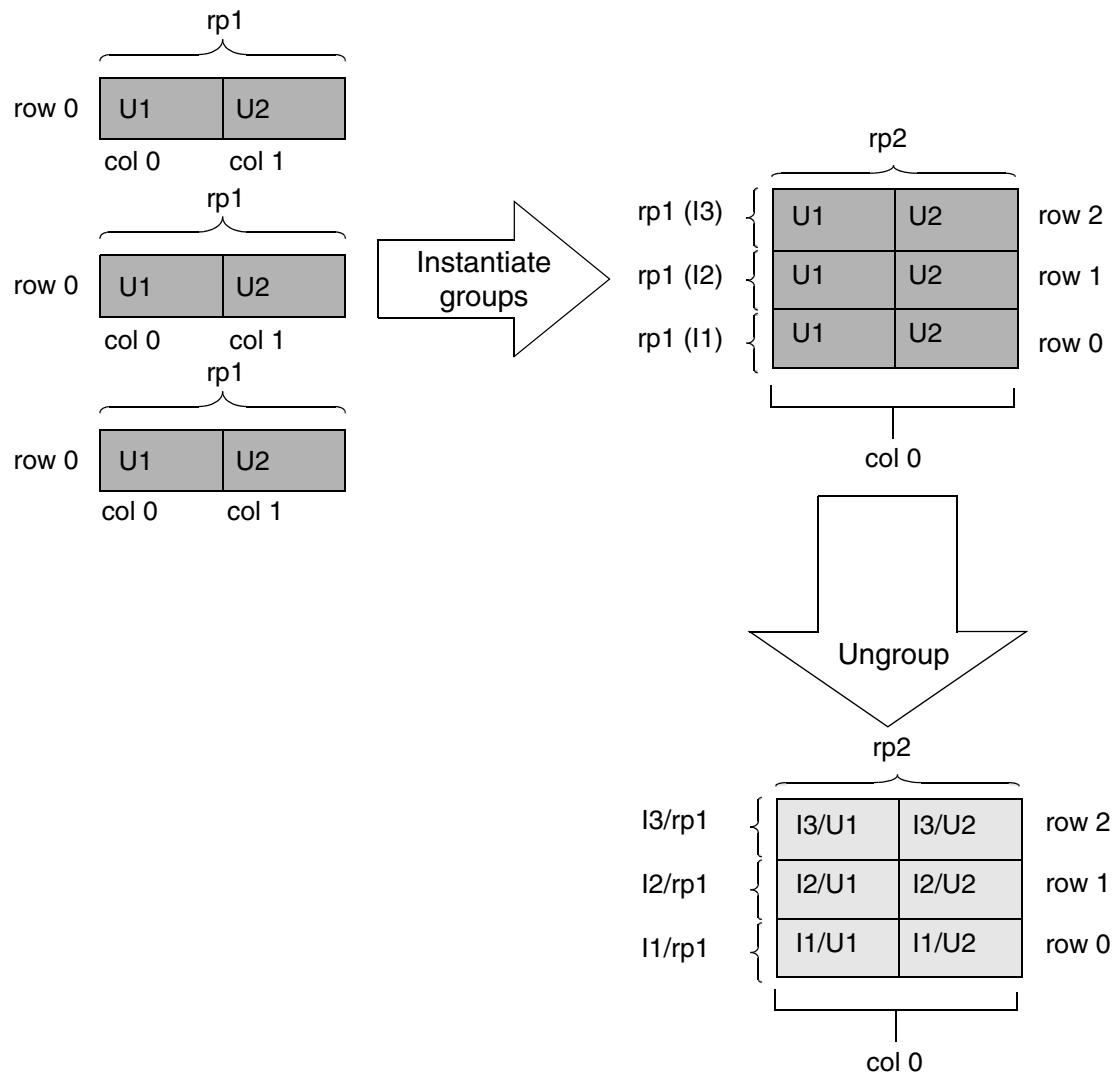
```
create_rp_group I1/rp1 -design mid_design -columns 2 -rows 1
  add_to_rp_group mid_design::I1/rp1 -leaf I1/U1 -column 0 -row 0
  add_to_rp_group mid_design::I1/rp1 -leaf I1/U2 -column 1 -row 0

create_rp_group I2/rp1 -design mid_design -columns 2 -rows 1
  add_to_rp_group mid_design::I2/rp1 -leaf I2/U1 -column 0 -row 0
  add_to_rp_group mid_design::I2/rp1 -leaf I2/U2 -column 1 -row 0

create_rp_group I3/rp1 -design mid_design -columns 2 -rows 1
  add_to_rp_group mid_design::I3/rp1 -leaf I3/U1 -column 0 -row 0
  add_to_rp_group mid_design::I3/rp1 -leaf I3/U2 -column 1 -row 0

create_rp_group rp2 -design mid_design -columns 1 -rows 3
  add_to_rp_group mid_design::rp2 -hierarchy mid_design::I1/rp1 -column 0 -row 0
  add_to_rp_group mid_design::rp2 -hierarchy mid_design::I2/rp1 -column 0 -row 1
  add_to_rp_group mid_design::rp2 -hierarchy mid_design::I3/rp1 -column 0 -row 2
```

Figure 9-33 Instantiating Groups in a Hierarchical Group



Effect of Uniquifying on Hierarchical Relative Placement

The `uniquify_fp_mw_cel` command can change each instantiation of hierarchical relative placement structures.

The following example shows a typical uniquifying flow:

```
...
icc_shell> read_verilog my_non_uniquified_design.v
icc_shell> source my_rp_constraint.tcl
icc_shell> uniquify_fp_mw_cel
icc_shell> read_def my_floorplan.def
icc_shell> create_placement
or
icc_shell> place_opt
...
```

For example, uniquifying

```
create_rp_group grp_ripple -design ripple
...
create_rp_group grp_top -design top -columns 1 -rows 2
add_to_rp_group top::grp_top -hierarchy ripple::grp_ripple -instance u1 \
    -column 0 -row 0
add_to_rp_group top::grp_top -hierarchy ripple::grp_ripple -instance u2 \
    -column 0 -row 1
```

results in

```
create_rp_group grp_ripple -design ripple
...
create_rp_group grp_ripple -design ripple_0
...
create_rp_group grp_top -design top -columns 1 -rows 2
add_to_rp_group top::grp_top -hierarchy ripple::grp_ripple \
    -instance u1 -column 0 -row 0
add_to_rp_group top::grp_top -hierarchy ripple_0::grp_ripple \
    -instance u2 -column 0 -row 1
```

Adding Keepouts

You can add soft, hard, or space keepouts within relative placement groups either in the relative placement hierarchy browser or from the command line.

All hard, soft, and space keepouts prevent the placement of relative placement cells in that location during both placement and legalization. The differences between the three keepout types are

- How the nonrelative placement cells are handled during legalization

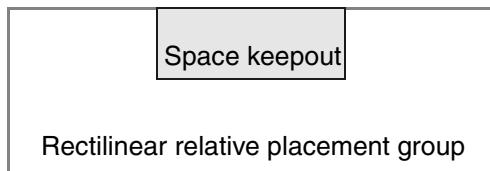
Nonrelative placement cells can be placed in both soft and space keepouts, but not in hard keepouts.

- How the nonrelative placement cells are handled during placement

Nonrelative placement cells can be placed in space keepouts, but not in hard or soft keepouts. To preserve space for nonrelative placement cells inside a relative placement group, such as buffers and control flip-flops, choose the space keepout type.

To fill empty space in rectilinear relative placement groups with nonrelative placement cells, you can create space keepouts to place the nonrelative placement cells in the rectilinear region as shown in [Figure 9-34](#).

Figure 9-34 Space Keepout in a Rectilinear Relative Placement Group



- How the keepout width and height are handled during optimization

For hard and space keepouts, the width and height do not change, although the location might, as a result of optimization performed on the relative placement group. Upsizing of relative placement cells can cause an increase column width.

For soft keepouts, the width and height might change as a result of optimization performed on the relative placement group. The column width is maintained, even during upsizing of relative placement cells.

- Their ability to place keepouts over tap cells

Soft keepouts can be placed over tap cells.

By default, hard and space keepouts cannot be placed over tap cells.

To allow placement of hard and space keepouts over tap cells for a relative placement group, use the `-allow_keepout_over_tapcell true` option of the `create_rp_group` or `set_rp_group_options` command.

When you specify keepouts, keep the following points in mind:

- You can specify whether the keepout is a hard, soft, or space keepout.
If you do not specify the keepout type, the IC Compiler tool creates a hard keepout.
- You can specify the width and height of a keepout.
 - The unit of width for a keepout is the width of a site row. If you do not specify the width, it defaults to the width of the column into which you are adding the keepout.
 - The unit of height for a keepout is the height of a site row. If you do not specify the height, it defaults to the height of the row into which you are adding the keepout.
- You can add a cell, a hard keepout, or a soft keepout, but not a space keepout at the same position in the relative placement group.

When a cell and a keepout occupy the same position, the cell is placed first and the keepout is placed to the right of the cell.

To add keepouts in the relative placement hierarchy browser (Placement > New RP Hierarchy View),

1. Select the group you want to add to in the left pane.
2. Select the Grid tab in the right pane.
3. Select a location in the grid view.
Keepouts can be added in open or occupied locations.
4. Right-click and choose Add Object.
The Add Objects to RP dialog box appears.
5. Specify the keepout name, width, height, and type.
6. Click OK.

Alternatively, you can use the `add_to_rp_group` command.

For example, to create a hard keepout named gap1, enter

```
icc_shell> add_to_rp_group TOP::misc -keepout gap1 \
    -column 0 -row 2 -width 15 -height 1
```

You can also add keepouts by using the relative placement editing dialog box in the GUI. For more information, see “[Editing and Creating Relative Placement Groups](#)” on page 9-73.

Placement and Optimization of Relative Placement Groups

During placement and optimization, the structure of the relative placement groups is preserved and the cells in each group are placed as a single entity. The following sections describe these aspects of relative placement:

- [Constraining the Size of a Relative Placement Group for Optimization](#)
 - [Performing Relative Placement in a Design Containing Tap Cells](#)
 - [Handling Fixed Cells During Relative Placement](#)
 - [Handling Physical-Only Cells During Relative Placement](#)
 - [Using Move Bounds to Constrain Relative Placement](#)
 - [Supporting Relative Placement Groups in Virtual Flat Placement](#)
 - [Propagating Relative Placement Groups in Physical Hierarchy](#)
-

Constraining the Size of a Relative Placement Group for Optimization

During optimization, if the tool resizes the cells in a relative placement group, the size of the group might increase. You can constrain the size of a relative placement group by specifying a maximum height and width by using the `-max_rp_height` and `-max_rp_width` options of the `create_rp_group` or `set_rp_group_options` command.

The tool honors the maximum height and width constraints you specify when you perform optimization by using the `place_opt`, `psynopt`, `clock_opt`, and `route_opt` commands.

You can remove the maximum height and width constraints by using the `remove_rp_group_options` command.

Performing Relative Placement in a Design Containing Obstructions

During placement, relative placement groups avoid placement blockages (obstructions) that are defined in the DEF file or created by the `create_placement_blockage` command. A relative placement group can be broken into pieces that straddle obstructions, yet maintain the relative placement structure.

If the height of the obstruction is below a certain threshold, the relative placement cells are shifted vertically; otherwise, the relative placement cells are shifted horizontally.

[Figure 9-35](#) shows the placement of relative placement cells in a design containing obstructions that were either defined in the DEF file or created by `create_placement_blockage`. The obstruction in columns one and two is below the threshold, so the tool shifts the cells vertically. The obstruction in column four is greater than the threshold, so the tool shifts the cells horizontally.

Figure 9-35 Relative Placement in a Design Containing Obstructions

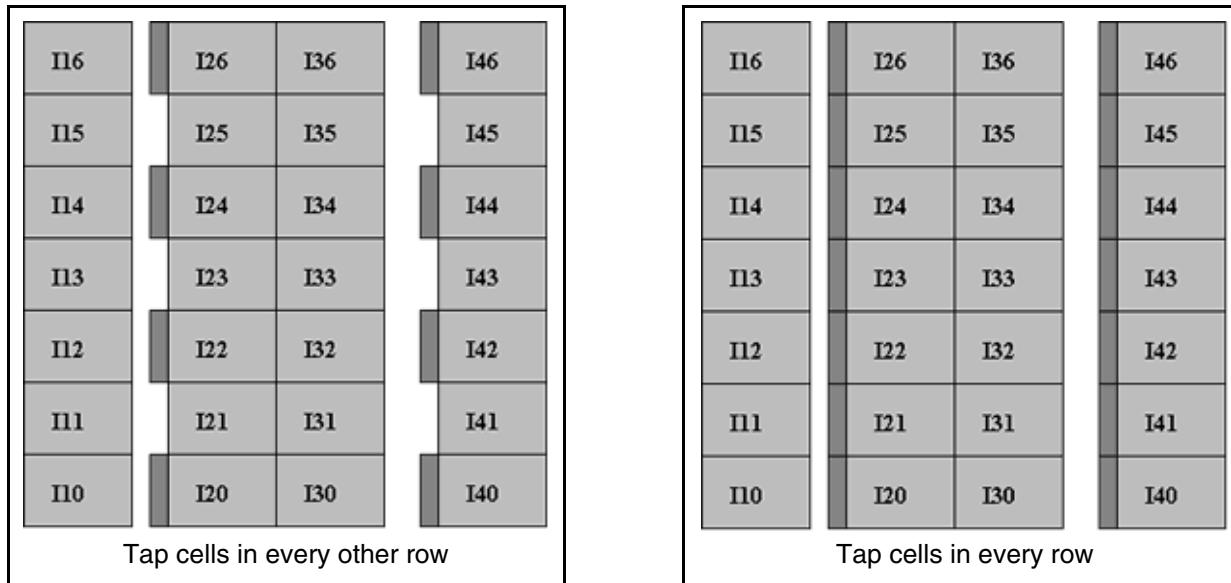
	1 4	2 4		
row 4	0 4	1 3	2 3	3 4
row 3	0 3	1 2	2 2	3 3
row 2	0 2	Obstruction		3 2
row 1	0 1	1 1	2 1	3 1
row 0	0 0	1 0	2 0	3 0
	col 0	col 1	col 2	col 3
				col 4

Performing Relative Placement in a Design Containing Tap Cells

Tap cells are special nonlogic cells with well and substrate ties. Tap cells can be defined in the DEF file or created by a tap cell command such as the `add_tap_cell_array` command. For more information about tap cells, see [Chapter 7, “Chip Finishing and Design for Manufacturing.”](#)

During placement, the cells or columns of a relative placement group are automatically shifted to avoid tap cells while maintaining the relative positions of the cells in the relative placement group.

[Figure 9-36](#) shows how placement of a relative placement group is affected by a tap cell array. The design on the left contains tap cells in every other row, and the design on the right contains tap cells in every row. In both cases, the columns of the relative placement group are shifted to avoid the tap cells but the alignment of the relative placement group is maintained.

Figure 9-36 Shifting Columns to Avoid Tap Cell Array

Handling Fixed Cells During Relative Placement

To specify how to treat fixed cells in the floorplan during legalization of relative placement groups, use the `-place_around_fixed_cells` option of the `create_rp_group` or `set_rp_group_options` command. This option applies to top-level relative placement groups but not to hierarchical relative placement groups.

[Table 9-2](#) shows how the placer handles fixed cells when you specify the `-place_around_fixed_cells` option with a different keyword.

Table 9-2 Keywords for the -place_around_fixed_cells Option

To do this	Use this keyword
Legalize relative placement groups around fixed standard cells and avoid fixed physical-only cells.	standard
Legalize relative placement groups around fixed physical-only cells and avoid fixed standard cells.	physical_only
Legalize relative placement groups around both fixed standard cells and fixed physical-only cells. This is the default.	all
Avoid both fixed standard cells and fixed physical-only cells.	none

The following command places the rp relative placement group around fixed standard cells but avoids fixed physical-only cells during legalization.

```
icc_shell> create_rp_group rp -design d1 -rows 2 -columns 2 \
           -place_around_fixed_cells standard
```

The following command avoids both fixed standard cells and fixed physical-only cells during the legalization of the d1::rp relative placement group.

```
icc_shell> set_rp_group_options d1::rp -place_around_fixed_cells none
```

Handling Physical-Only Cells During Relative Placement

Physical-only cells are nonlogic cells that have only PG pins or no pin at all, such as tap cells and decoupling capacitors. During the placement of relative placement groups, physical-only cells are not recognized and can cause possible overlapping with relative placement groups. By default, the alignment of relative placement groups is preserved during placement and legalization. However, the alignment of relative placement groups that are overlapped by physical-only cells is often disturbed after the `legalize_placement` step.

For better alignment, the IC Compiler tool allows both physical-only cells and relative placement cells in a relative placement group by default. You can place physical-only cells just like logic cells by specifying the alignment methods, cell orientation, anchor location, bit-stack placement, and so forth.

For example, to add a physical-only cell named PO_cell to a relative placement group, enter

```
icc_shell> add_to_rp_group design::rp_group -leaf PO_cell \
           -column 0 -row 0
```

To remove the PO_cell from the relative placement group, enter

```
icc_shell> remove_from_rp_group design::rp_group -leaf PO_cell
```

To return a collection of physical-only cells in a relative placement group, use the `rp_group_references -physical_only` command.

Note that the `extract_rp_group` command does not support physical-only cells.

Using Move Bounds to Constrain Relative Placement

You can constrain the placement of relative placement cells by defining move bounds with fixed coordinates (relative placement does not support group bounds). Both soft bounds and hard bounds are supported for relative placement cells, and both rectangular bounds and rectilinear bounds are supported.

Warning:

If you use move bounds to constrain relative placement, all cells in a relative placement group must be in the same move bound; otherwise, the tool issues an error message and ignores the relative placement constraints.

To constrain relative placement by using move bounds, use the `create_bounds` command and specify the individual cell names as provided in an `add_to_rp_group` command as the command argument. You cannot specify a relative placement group as the command argument.

For example, enter

```
icc_shell> create_bounds -coordinate {100 100 200 200} \
    "U1 U2 U3 U4" -name bound1
```

Supporting Relative Placement Groups in Virtual Flat Placement

You can place relative placement groups during virtual flat placement in the design planning flow. To achieve better correlation between the `create_fp_placement` and `create_placement` commands, the `create_fp_placement -no_legalize` command supports three types of relative placement groups that contain macros only, standard cells only, and both macros and standard cells.

During the design planning flow, you can perform the following relative placement functions:

- Specify bit-slice placement, compression, or vertical compression by using the `-placement_type` option with the `bit_slice`, `compression`, or `vertical_compression` keyword respectively.
- Specify the anchor location by using the `-x_offset` and `-y_offset` options.
- Specify the utilization percentage by using the `-utilization` option.
- Ignore the relative placement group by using the `-ignore` option.

However, you should not use the following relative placement functions:

- Specify the group alignment pin by using the `-pin_align_name` option.
- Specify the right alignment by using the `-alignment bottom_right` option.

- Allow keepouts over tap cells by using the `-allow_keepout_over_tapcell` option.
- Specify the orientation of relative placement groups by using the `-group_orient` option.

For designs containing relative placement groups that have either only macros or standard cells, use the following flow:

1. Define the relative placement constraints.
 - Create the relative placement groups by using the `create_rp_group` command.
 - Add relative placement objects to the groups by using the `add_to_rp_group` command.
2. Run virtual flat placement by using the `create_fp_placement -no_legalize` command.
3. Mark all macros with the `is_fixed` attribute.
4. Legalize your design by using the `legalize_placement` command.
5. Mark all relative placement cells with the `is_fixed` attribute.
6. Perform physical placement by using the `place_opt` command.

You can skip Steps 4 and 5 if the relative placement group contains standard cells only.

For designs containing relative placement groups that have mixed macros and standard cells, the flow to perform virtual flat placement follows these steps:

1. Run virtual flat placement by using the `create_fp_placement -no_legalize` command.
2. Mark all macros with the `is_fixed` attribute.
3. Legalize your design by using the `legalize_placement` command.
4. Mark all macros and standard cells in the relative placement groups with the `is_fixed` attribute.
5. Continue the design planning flow.

For more information about performing relative placement in the design planning flow, see the virtual flat placement information in the *IC Compiler Design Planning User Guide*.

Propagating Relative Placement Groups in Physical Hierarchy

During the commit and uncommit hierarchy steps of the design planning flow, the `commit_fp_plan_groups` and `uncommit_fp_soft_macros` commands support relative placement groups, including hierarchical relative placement groups. Before you run the `commit_fp_plan_groups` command, you should perform initial virtual flat placement. For more information, see “[Supporting Relative Placement Groups in Virtual Flat Placement](#)” on page 9-55.

The `commit_fp_plan_groups` command converts plan groups into new soft macros. If the plan group includes relative placement groups, they are propagated to soft macros. The `uncommit_fp_soft_macros` command converts soft macros into top-level plan groups. If the soft macro includes relative placement groups, they are propagated to the plan group in the top level.

Use the following flow for designs containing relative placement groups:

1. Place and legalize the design with relative placement groups by using the `create_fp_placement -no_legalize` and `legalize_placement` commands.
During the `legalize_placement` step, the relative placement groups remain intact inside the plan groups that they belong.
2. Convert plan groups into soft macros, also called a child cell or a physical block, by using the `commit_fp_plan_groups` command.
The command removes relative placement groups that belong to the plan group from the top CEL view and pushes them down to soft macros. Each soft macro is a CEL view of the block in the same Milkyway design library.
3. Fix the locations of the relative placement groups that include macros by setting the `is_fixed` attribute.
4. Perform place and route on each soft macro independently.
5. (Optional) Convert all soft macros into plan groups by using the `uncommit_fp_soft_macros` command.
During uncommit hierarchy, the soft macro child cell instances are pushed back up to the top CEL view.
6. (Optional) Validate the relative placement groups by performing the following commands.

```
write_rp_groups -all -nosplit -output afu.tcl  
check_rp_groups -all -verbose  
report_rp_group_options
```

Note that if you need to place the design after the design planning flow, both the `create_placement` and `refine_placement` commands respect relative placement groups that are inside plan groups during placement.

For more information, see the relative placement information in the *IC Compiler Design Planning User Guide*.

Postplacement for Relative Placement Groups

After placement, you can add incremental relative placement groups to a placed design without disturbing the initial placement. You can also improve the quality of results in relative placement by controlling movement, legalizing relative placement groups, and allowing nonrelative placement cells in relative placement groups. The following sections describe how to perform these tasks.

- [Adding Incremental Relative Placement Groups](#)
 - [Controlling Movement When Legalizing Relative Placement Groups](#)
 - [Ignoring Site Rows When Legalizing Relative Placement Groups](#)
 - [Legalizing Relative Placement Groups in a Placed Design](#)
 - [Exposing Unused Space in Relative Placement Groups](#)
-

Adding Incremental Relative Placement Groups

You can create relative placement groups of cells in an already placed design and place the relative placement groups incrementally where the `refine_placement` command does not disturb the initial placement significantly. Note that if you select cells that are placed far apart in the initial placement for the same relative placement group, performing incremental relative placement might degrade the QoR.

The following script shows an example of how to add incremental relative placement groups:

```
icc_shell> create_placement
icc_shell> legalize_placement
icc_shell> create_rp_group new_rp -design design1 -columns 1 -rows 4
icc_shell> add_to_rp_group design1::new_rp -leaf U1 -column 0 -row 0
icc_shell> add_to_rp_group design1::new_rp -leaf U2 -column 0 -row 1
...
icc_shell> refine_placement
```

For more information about the `refine_placement` command, see “[Refining Placement](#)” on page [4-90](#).

For a placed design, if you create a new relative placement group using the `-x_offset` and `-y_offset` options and then run `legalize_placement`, the tool places and legalizes the newly created group anchored by the specified x- and y-coordinates.

Controlling Movement When Legalizing Relative Placement Groups

You can control the movement of a relative placement group during legalization by using the `-move_effort` option of the `create_rp_group` command. During coarse placement, the tool estimates an initial location for every top-level relative placement group. The `-move_effort` option controls the extent to which a relative placement group can be moved from its initial location to preserve the relative placement without violating relative placement constraints. When you change the option setting from a higher effort level to a lower effort level, you reduce the size of the region searched for placement of a relative placement group.

For example, to keep the relative placement group close to the initial location estimated by the tool during coarse placement, enter

```
icc_shell> create_rp_group grp_ripple -design ripple -move_effort low
```

The `set_rp_group_options` command also supports the `-move_effort` option.

Ignoring Site Rows When Legalizing Relative Placement Groups

You can ignore specific site rows in your floorplan when legalizing a relative placement group by using the `-ignore_rows` option with the `create_rp_group` or `set_rp_group_options` command. Identify the site rows to be ignored by applying the `rp_ignore_row_name` attribute by using the `set_attribute` command.

The following example shows how you can prevent cells from the TOP::RP1 relative placement group from being placed on the PG site row during legalization:

```
icc_shell> define_user_attribute -type string -class site_row \
   rp_ignore_row_name
icc_shell> set_attribute [get_site_row PG] rp_ignore_row_name PG
icc_shell> set_rp_group_options TOP::RP1 -ignore_rows PG
```

Use the `remove_rp_group_options -ignore_rows` command to remove the settings specified by using the `-ignore_rows` option with the `create_rp_group` or `set_rp_group_options` command.

Legalizing Relative Placement Groups in a Placed Design

You can improve the placement of relative placement groups in an already placed design by legalizing only the relative placement groups. To legalize the detailed placement of relative placement groups but not nonrelative placement cells, use the `legalize_rp_placement` command.

The following command legalizes all relative placement groups in the design.

```
icc_shell> legalize_rp_placement
```

You can also specify a list of relative placement groups to be legalized. For example, the following command legalizes the TOP::RP1 and TOP::RP2 relative placement groups.

```
icc_shell> legalize_rp_placement {TOP::RP1 TOP::RP2}
```

Using the `legalize_rp_placement` command for incremental relative placement groups reduces the turnaround time. The following script shows an example of how to legalize incremental relative placement groups with this command.

```
...
#Create RP1 and add cells to RP1
create_rp_group RP1 -design TOP ...
add_to_rp_group TOP::RP1 ...

#Create RP2 and add cells to RP2
create_rp_group RP2 -design TOP ...
add_to_rp_group TOP::RP2 ...

#Run placement using create_placement, refine_placement, or place_opt
create_placement
...
#Create RP3 and add cells to RP3
create_rp_group RP3 -design TOP ...
add_to_rp_group TOP::RP3 ...

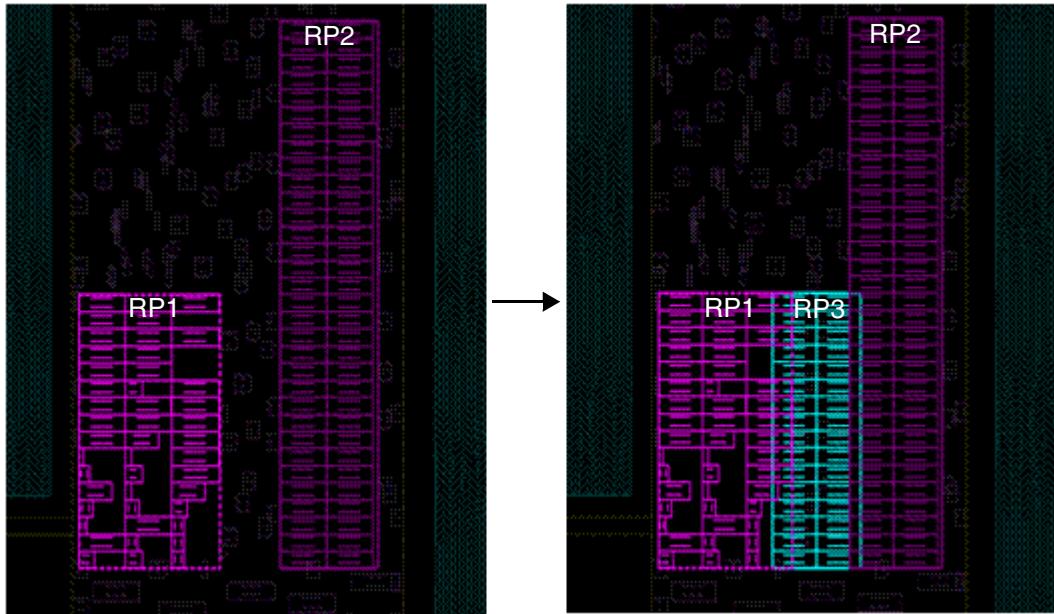
#Legalize relative placement groups using legalize_rp_placement or
#legalize_rp_placement -legalize_over_rp
legalize_rp_placement -legalize_over_rp RP3
legalize_placement
...
```

You can specify relative placement groups to be legalized over other unspecified relative placement groups by using the `-legalize_over_rp` option with the `legalize_rp_placement` command. Before running the `legalize_rp_placement` command, you need to perform an initial placement.

For example, the following command legalizes the RP3 relative placement group over the RP1 and RP2 relative placement groups. [Figure 9-37](#) shows the placement before and after running the command.

```
icc_shell> legalize_rp_placement -legalize_over_rp RP3
```

Figure 9-37 Legalizing the RP3 Relative Placement Group Over Other Groups



Because the command legalizes relative placement groups only, you might need to run the `legalize_placement` command to legalize nonrelative placement cells after running the `legalize_rp_placement` command. You should also legalize the overlapping nonrelative placement cells and relative placement groups, if any. In this example, the RP1 and RP2 relative placement groups are overlapped by the RP3 relative placement group.

You might need to legalize the RP1 and RP2 relative placement groups by performing the following tasks in sequence:

1. Set new anchor points by using the `-x_offset` and `-y_offset` options or control the movement by using the `-move_effort` option of the `set_rp_group_options` command.
2. Run the `legalize_rp_placement {TOP::RP1 TOP::RP2}` or `legalize_placement` command.

Exposing Unused Space in Relative Placement Groups

To allow the placer to use unused space in relative placement groups for nonrelative placement cells, you can specify the `-allow_non_rp_cells` option of the `create_rp_group` or `set_rp_group_options` command. When this option is set, the `refine_placement` and `place_opt` commands fix the relative placement cells in the specified relative placement groups so that the unused space is shown. During placement optimization, the coarse placer can consider the unused space for nonrelative placement cells to reduce congestion and improve QoR. This option applies to top-level relative placement groups, but not to hierarchical relative placement groups.

To reset this option setting, specify the `-allow_non_rp_cells` option of the `remove_rp_group_options` command.

Postplacement Optimization of Relative Placement Groups

During postplacement optimization, relative placement cells can be modified, moved, or removed, and buffering can occur. The following sections describe the preservation, buffering strategy, and optimization for relative placement groups:

- [Preserving Relative Placement Structures](#)
- [Optimizing Relative Placement Cells](#)
- [Buffering Strategy for Relative Placement Groups](#)

Preserving Relative Placement Structures

When a relative placement cell is modified or moved, the relative placement structure can be disturbed. When a relative placement cell is removed during optimization, the relative placement information attached to the instance is also removed, disrupting the relative placement structure.

To preserve the relative placement structures during various postplacement optimization processes, use the `create_rp_group` or `set_rp_group_options` command to specify the `fixed_placement` or `size_only` keyword with the appropriate option as shown in [Table 9-3](#). The `fixed_placement` attribute keeps the relative placement structures fixed. The `size_only` attribute prevents relative placement cells from being removed but allows cell sizing during optimization.

For clock tree synthesis and physical optimization, you specify the preservation attribute by using the `fixed_placement` or `size_only` keyword. For routing optimization, you specify the preservation attribute by using the `fixed_placement` or `in_place_size_only` keyword.

Table 9-3 shows the options that mark the relative placement cells as `fixed_placement`, `size_only`, or `in_place_size_only` during the affected commands.

Table 9-3 Options to Preserve Relative Placement Structures

Process	Option	Affected commands
Clock tree synthesis	<code>-cts_option</code>	<code>clock_opt</code> <code>compile_clock_tree</code> <code>optimize_clock_tree</code>
Physical optimization	<code>-psynopt_option</code>	<code>psynopt</code> <code>clock_opt -only_psyn</code>
Routing	<code>-route_opt_option</code>	<code>route_opt</code>

By default, the tool allows relative placement cells to be sized during

- The `psynopt` and `place_opt` steps.
- The embedded `psynopt` step of the `clock_opt -only_psyn` command.

If relative placement cells are upsized or downsized, the relative placement cell alignment is maintained for placement.

To set the `size_only` attribute on all relative placement cells during physical optimization, enter

```
icc_shell> set_rp_group_options -psynopt_option size_only \
[get_rp_groups *]
```

To size the relative placement cells during clock tree synthesis and physical optimization, enter

```
icc_shell> set_rp_group_options -cts_option size_only \
-psynopt_option size_only [get_rp_groups *]
```

You can reduce waste space in relative placement groups by specifying the `-cts_option size_only` option during the `clock_opt` step or during clock tree synthesis and optimization. You use this option only when there is enough space for sizing.

To size the relative placement cells during postroute optimization, enter

```
icc_shell> set_rp_group_options -route_opt_option in_place_size_only \
[get_rp_groups *]
```

Optimizing Relative Placement Cells

To allow all optimizations on relative placement cells during placement performed by the `psynopt` and `place_opt` commands, specify the `all_optimization` keyword of the `-psynopt_option` option when you run the `create_rp_group` or `set_rp_group_options` command. To preserve relative placement cells during physical optimization, specify the `fixed_placement` or `size_only` keyword of the `-psynopt_option` option. For more information about preserving relative placement cells, see “[Preserving Relative Placement Structures](#)” on page 9-62.

When you specify the `-psynopt_option all_optimization` option, the tool performs all optimizations on the relative placement cells in the specified relative placement groups. After the optimizations, the relative placement cells remain inside the relative placement groups and the structures of the relative placement groups are preserved. However, some empty areas in the relative placement groups might occur because of the removal of relative placement cells. You should check whether removing the relative placement cells is acceptable for the specified relative placement groups on which the tool performs all optimizations.

Buffering Strategy for Relative Placement Groups

By default, optimization performed by the `psynopt` or `route_opt` command can insert buffers inside relative placement groups or into relative placement nets (nets that connect relative placement cells within a relative placement group or nets that connect two lower-level hierarchical groups in the same top-level relative placement group). These inserted buffers can increase congestion and degrade the quality of results (QoR).

To avoid this situation, you can use the `set_rp_group_options` command to specify a buffering strategy for relative placement groups.

- To prevent the insertion of buffers inside the areas of placed relative placement groups, specify the `-auto_blockage` option.
- To prevent the insertion of buffers into relative placement nets, specify the `-disable_buffering` option.

For example, the following command prevents nonrelative placement cells from being placed inside the areas of relative placement groups and prevents buffers from being inserted into relative placement nets.

```
icc_shell> set_rp_group_options -auto_blockage \
           -disable_buffering [get_rp_groups *]
```

Analyzing the Relative Placement Results

The following sections explain methods for analyzing your relative placement results:

- [Checking Relative Placement Constraints](#)
 - [Locating a Relative Placement Group](#)
 - [Analyzing Relative Placement Groups in the GUI](#)
-

Checking Relative Placement Constraints

To check whether the relative placement constraints have been met, run the `check_rp_groups` command.

The `check_rp_groups` command checks for the following failures:

- One or more cells in the relative placement group cannot be legally placed.
- The relative placement group cannot be placed as a whole.
- The height or width of the relative placement group is greater than the height or width of the core area.
- The user-specified orientation cannot be met.

The generated report contains separate sections for critical and noncritical failures. If the failure does not prevent placement but causes relative placement constraints to be violated, such as an alignment, utilization, or orientation failure, the failure is considered noncritical. If a failure prevents the group from being placed as a single entity that is defined by the relative placement constraints, the failure is considered a critical failure. To check the specified relative placement groups that cannot be placed, use the `-critical` option.

You can check all relative placement groups by specifying the `-all` option, or you can specify a list of relative placement groups to check. For example,

```
icc_shell> check_rp_groups -all
...
*****
RP GROUP: ORCA::top
-----
Warning: The alignment has not been respected for RP group
ORCA::top
```

To report more details about failures in cell orientation, alignment, and utilization with the exact locations, use the `-verbose` option.

The following example shows a report that contains an alignment failure:

```
icc_shell> check_rp_groups -all -verbose
...
*****
RP GROUP: ORCA::top
-----
Warning: The alignment has not been respected for RP group ORCA::top

Location details of alignment failure inside RP Group ORCA::top:
-----
RP Group           Row  Column  Failed Alignment
-----
example3::gp_1_in_example3  1    0        bottom-right
                                         (specified on RP cell U18)
ORCA::top          1    1        bottom-left
-----
(RPGP-038)
{ORCA::top}
```

The following example shows a report that contains a utilization failure:

```
icc_shell> check_rp_groups -all -verbose
...
RP GROUP: ORCA::top
-----
Warning: The utilization has not been respected for the RP group
ORCA::top

Utilization is not met at the following columns inside RP Group
ORCA::top:
-----
RP Group           Column
-----
ORCA::top          1
ORCA::top          0
-----
(RPGP-039)
{ORCA::top}
```

By default, the report is output to the screen. To save the report to a file, specify the file name by using the **-output** option. For example, to check the relative placement constraints for

the compare17::seg7 and compare17::rp_group3 relative placement groups and to save the output in a file named rp_failures.log, use the following command:

```
icc_shell> check_rp_groups "compare17::seg7 compare17::rp_group3" \
-output rp_failures.log
...
*****
RP GROUP: compare17::seg7
-----
Warning: Possible placement failure of relative placement group
'compare17::seg7'. (RPGP-027)
...
...
*****
RP GROUP: compare17::rp_group3
-----
Warning: The utilization has not been respected for the RP group
'compare17::rp_group3'. (RPGP-039)
```

You can also check the relative placement constraints in the GUI by choosing Placement > Check RP Groups. When you check the constraints in the GUI, all relative placement groups are checked. In the generated report, critical failures are indicated by an exclamation mark (!). When you select reported violations, the location of the violation is displayed in the layout view.

Note that the `check_rp_groups` command does not check for the failure that keepouts are not created correctly in the relative placement group.

Checking Relative Placement Groups Before Placement

Before you run placement, use the `check_rp_groups` command to check for relative placement issues that might lead to critical or noncritical failures after placement.

The following example shows a report for a relative placement group that contains cells with the `is_fixed` attribute:

```
icc_shell> check_rp_groups -all
...
*****
RP GROUP: ORCA::top
-----
Error: Fixed attribute is set on some cells of relative placement group
ORCA::top (RPGP-122)
{ORCA::top}
```

The following example shows a report for a relative placement group that contains cells with conflicting move bound constraints:

```
icc_shell> check_rp_groups -all
...
*****
RP GROUP: ORCA::top
-----
Warning: The relative placement constraints cannot be honored for
relative placement group ORCA::top, because all cells of the relative
placement group do not belong to the same move bound. (RPGP-041)
{ORCA::top}
```

Locating a Relative Placement Group

You use the `get_location` command with its `-rp_group` option to get the grid location of a relative placement cell, group, or keepout.

The syntax is

```
get_location cell_or_group_list
    -rp_group containing_rp_group
```

The command returns a list of the x- and y-values of each object specified within the containing relative placement group that you specify.

If a group is instantiated multiple times within the group that contains it, a sublist of all locations is returned within the returned list of locations for each object.

If a group does not contain any of the objects, a message appears.

For example, to get the location of cell U11 in relative placement group `my_group`, enter the following commands:

```
icc_shell> set cell_rp_location \
    [get_location [get_cell U11] -rp_group my_group]
icc_shell> set x_rp_loc [lindex $cell_rp_location 0]
icc_shell> set y_rp_loc [lindex $cell_rp_location 1]
```

Analyzing Relative Placement Groups in the GUI

To analyze relative placement groups before placement, you can use the relative placement window in the GUI to check the hierarchy and net connections. For more information, see “[Using the Relative Placement Window](#)” on page 9-75.

Working With Relative Placement Groups in the GUI

The GUI provides the following methods of viewing and editing the relative placement groups:

- [Using the Relative Placement Hierarchy Browser](#)
 - [Viewing Relative Placement Groups](#)
 - [Viewing Relative Placement Group Net Connections](#)
 - [Viewing Net Connectivity Between Relative Placement Groups](#)
 - [Moving Relative Placement Groups](#)
 - [Editing and Creating Relative Placement Groups](#)
 - [Using the Relative Placement Window](#)
 - [Viewing the Hierarchy of Relative Placement Groups](#)
 - [Viewing Logical Connections in a Relative Placement Group](#)
 - [Editing Relative Placement Groups in a Relative Placement View Window](#)
 - [Editing the Structures of Relative Placement Groups](#)
-

Using the Relative Placement Hierarchy Browser

The relative placement hierarchy browser displays information about the relative placement groups in the current design.

To open the relative placement hierarchy browser, choose Placement > New RP Hierarchy View.

As shown in [Figure 9-38](#), the left pane of the relative placement hierarchy browser lists the relative placement groups in the current design. When you select a relative placement group in the left pane, its contents are listed in the right pane. When you select a group in the relative placement hierarchy browser in either the right or left pane, it is highlighted in the layout view.

Figure 9-38 Relative Placement Hierarchy Browser

The screenshot shows a software interface for managing relative placement groups. On the left, a tree view displays various components under the 'ORCA' design, including 'Crnt_Instrn_reg', 'Current_State_reg', 'Lachd_Result_reg', 'Oprnd_reg' (which is selected), 'PCint_reg', 'PopDataOut_reg', 's_op_reg', and 'Stack_Mem_reg'. Each component is associated with a specific number of rows and columns. On the right, a list panel titled 'full_name' shows two entries: 'Oprnd_A_reg' and 'Oprnd_B_reg', both with a value of 0.

name	design	rows	columns
ORCA::Crnt_Instrn_reg	ORCA	1	2
ORCA::Current_State_reg	ORCA	3	1
ORCA::Lachd_Result_reg	ORCA	16	1
ORCA::Oprnd_reg	ORCA	1	2
ORCA::PCint_reg	ORCA	8	1
ORCA::PopDataOut_reg	ORCA	11	1
ORCA::s_op_reg	ORCA	1	8
ORCA::Stack_Mem_reg	ORCA	1	8

full_name	r
ORCA::Oprnd_A_reg	0
ORCA::Oprnd_B_reg	0

Viewing Relative Placement Groups

Relative placement groups are visible and selectable in the layout view. In addition, the layout view provides a relative placement visual mode for displaying the relative placement groups and net connections.

To view relative placement groups in the visual mode,

1. Invoke relative placement visual mode by choosing Placement > Color By RP Groups.

The Visual Mode panel appears. The relative placement groups are listed by number, along with the number of cells in each group. If the design or the relative placement constraints have changed since you last invoked relative placement visual mode, this information might be out-of-date.

2. Update the relative placement information by clicking the Reload in the Visual Mode panel.

The Update Relative Placement Visual Mode dialog box appears.

3. Select “Color by RP groups” in the Update Relative Placement Visual Mode dialog box and click OK.

Each relative placement group is displayed in a different color in the layout view. If the design has not been placed, you do not see the relative placement groups in the layout view but the information in the Visual Mode panel is correct.

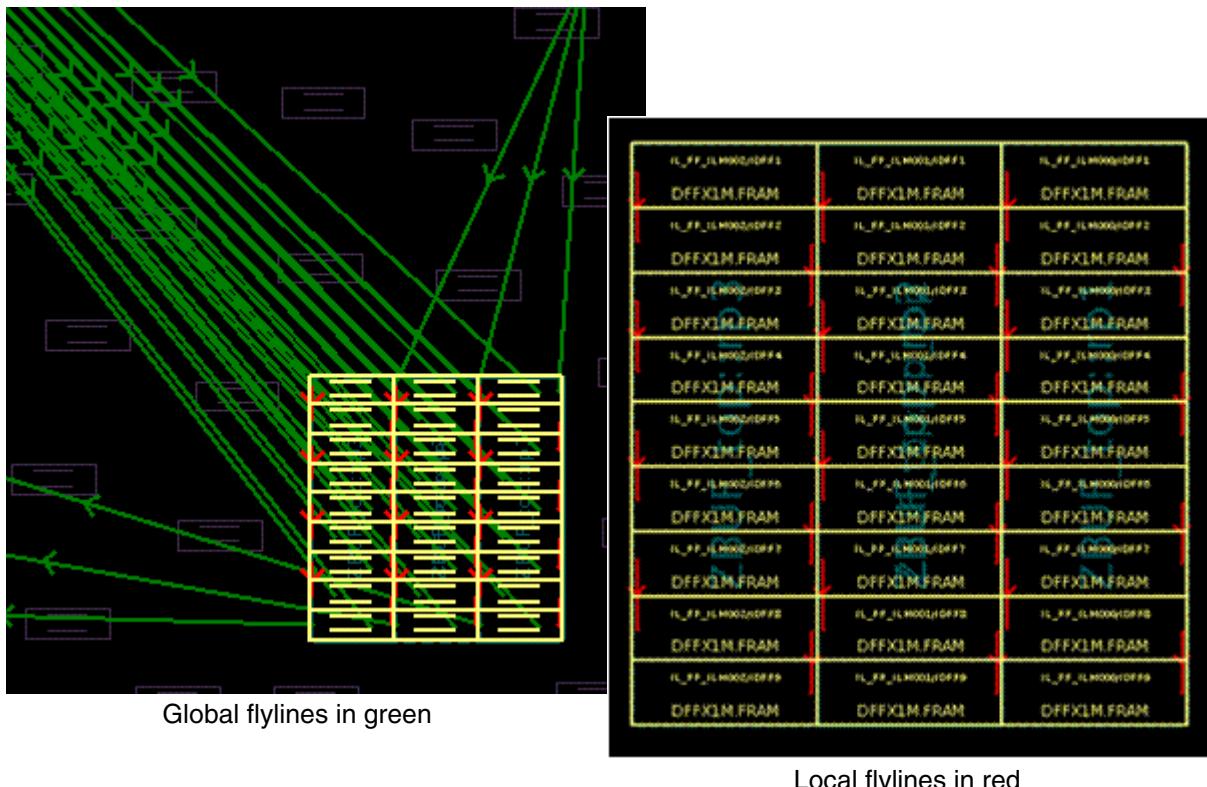
Viewing Relative Placement Group Net Connections

To view the net connections of a relative placement group in the visual mode,

1. Invoke relative placement visual mode by choosing Placement > Color By RP Net Connections.
2. Enter the relative placement group name in the Visual Mode panel.
3. Click the Reload button.

The RP Net Connections Visual Mode shows the number of pin-to-pin connections of the specified relative placement group. The pin-to-pin connections can be displayed in either flylines or routes. Two types of flyline connections that are highlighted in the visual mode view are called the local and global connections: the local is in red and the global is in green, as shown in [Figure 9-39](#). A histogram-like bar graph shows the number of connections for each type.

Figure 9-39 Net Connections in Visual Mode

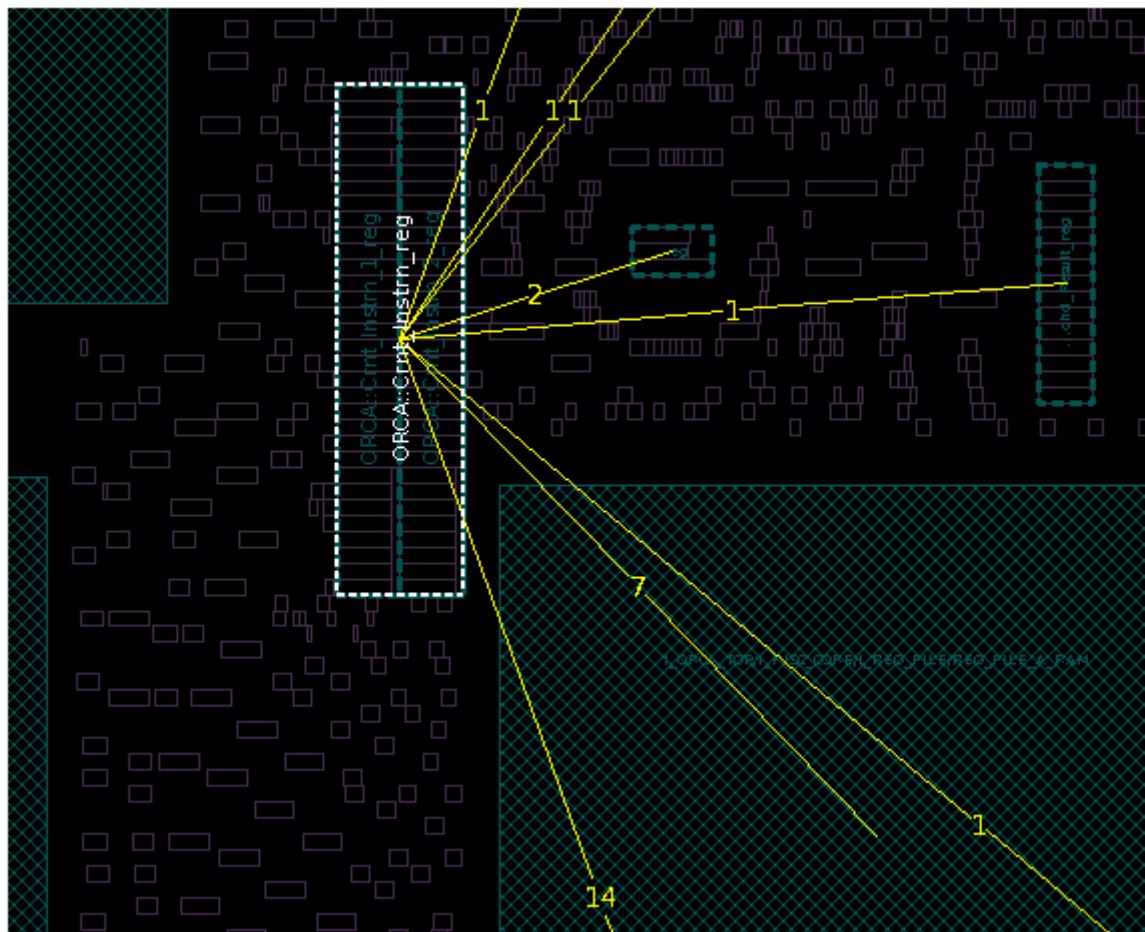


A dialog box that is displayed by clicking the List nets button lets you choose the local and global connections that are to be listed. You can also choose to display the local or global connections by checking the respective option in the Visual Mode panel.

Viewing Net Connectivity Between Relative Placement Groups

To generate a layout view showing the number of net connections between one selected relative placement group and other relative placement groups, macros, or plan groups, choose View > Net Connectivity in the GUI. An example of this is shown in [Figure 9-40](#).

Figure 9-40 Relative Placement Net Connectivity



Moving Relative Placement Groups

You can move relative placement groups in an active layout view by following these steps:

1. Click the  button on the Edit toolbar or choose Edit > Move/Resize.
2. Select all objects in a top-level relative placement group and drag them to a new location with the left mouse button.
Every object inside the selected relative placement group, including hierarchical relative placement groups, keepouts, and relative placement cells, is moved to the new location.
3. Run the `legalize_placement` command to legalize the relative placement group.

The following restrictions apply:

- You can move only top-level relative placement groups but not lower-level relative placement groups.
- You cannot move a relative placement group that contains relative placement cells marked as `is_fixed`.
- You cannot move a relative placement group that has an anchor point.

Editing and Creating Relative Placement Groups

To edit an existing relative placement group,

1. Use either of the following two ways to open an RP Editing dialog box:
 - From a layout window, choose Placement > Edit RP Group.
 - From a relative placement window, choose RP > Edit RP Group.The RP Editing dialog box appears.
2. Enter the relative placement group name in the RP group name box.

Alternatively, you can click the  button to display the pop-up menu. Select a relative placement group in the menu, and click OK.

3. Click the Edit option to edit an existing relative placement group.
If you want to create a new relative placement group, click the Create option.
4. Specify how many column and row numbers should be in the Columns and Rows boxes respectively.

5. Specify the object type by clicking the Cell, Keepout, or RP Group option.
If you click the Cell option, a dialog box appears.
6. (Optional) Set cell options in the dialog box before you select cells in the layout view or add cells manually in the table.
 - Choose the Cell options tab in the dialog box.
 - Choose the possible orientations and alignment methods for the selected cells.
7. (Optional) Set options in the dialog box to add cells.
Choose Add cells from selection tab in the dialog box to
 - Allow automatic expansion of row and column numbers for the relative placement group to accommodate all the selected cells.
This option is available only when you click the Create option. If you do not specify this option, the relative placement group is limited by the column and row numbers you specify in Step 4.
 - Add the selected cells in the table according to their physical locations.
 - Add the selected cells to the specified locations in the table.
 - Add the selected cells to the existing empty locations in the table.

Click the Add button in the dialog box to add cells.

Note:

When the number of the selected cells exceeds the available space in the table, it is undetermined which cells the tool chooses to put in the table.

8. Click the Add button.

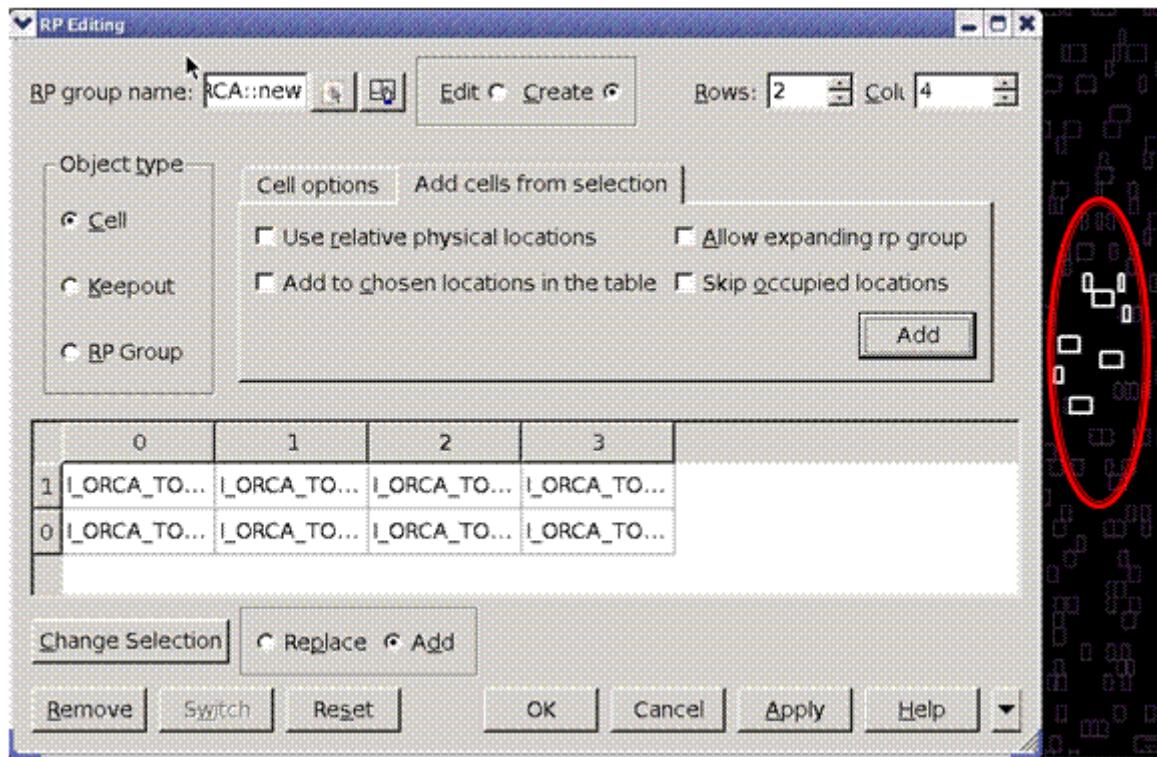
You can add objects to the table by entering the object names in the table.

9. Click OK or Apply.

You can remove objects from the table by clicking the Remove button. You can also drag an object from one location to another. To switch two selected objects between two columns or rows in the table, click the Switch button. To restart editing, click the Reset button. You can only add or create one relative placement group at any time by using the relative placement editing dialog box.

The relative placement editing dialog box allows you to cross-highlight objects between the table and layout view. [Figure 9-41](#) shows an example of using the relative placement editing dialog box to create a relative placement group of four columns and two rows. The layout view highlights the eight selected cells that are added to the relative placement group.

Figure 9-41 Relative Placement Editing Dialog Box



For more information about using the relative placement editing dialog box, see IC Compiler Help.

Alternatively, you can edit relative placement groups in a relative placement view window. See “[Editing Relative Placement Groups in a Relative Placement View Window](#)” on page 9-80 and “[Editing the Structures of Relative Placement Groups](#)” on page 9-81.

Using the Relative Placement Window

The relative placement window allows you to analyze the hierarchy and net connections of relative placement groups before running placement.

To open the relative placement window, you can choose one of the following two ways.

- From a top-level window, choose Window > New RP Window.
- From a layout window, choose Placement > New RP Window.

From a relative placement window, you can perform the following tasks:

- Edit existing relative placement groups or create new relative placement groups.
For more information, see “[Editing and Creating Relative Placement Groups](#)” on [page 9-73](#).
- Display the hierarchy of relative placement groups.
For more information, see “[Viewing the Hierarchy of Relative Placement Groups](#)” on [page 9-76](#).
- View net connections in a relative placement group.
For more information, see “[Viewing Logical Connections in a Relative Placement Group](#)” on [page 9-77](#).

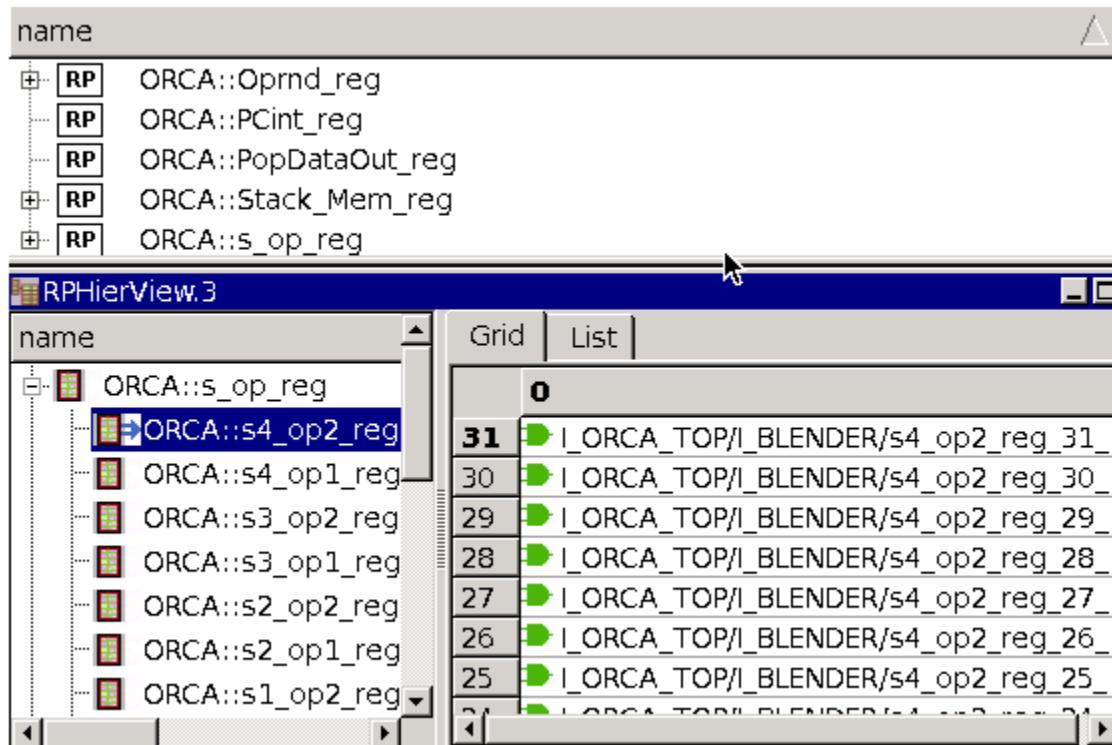
Viewing the Hierarchy of Relative Placement Groups

To view the hierarchy of relative placement groups, you can open a relative placement hierarchy view window by choosing one of the following two ways.

- From a relative placement window, choose View > RP Hierarchy Browser.
- From a layout window, choose Placement > New RP Hierarchy View.

[Figure 9-42](#) is an example that displays the hierarchy of relative placement groups.

Figure 9-42 Relative Placement Hierarchy View Window

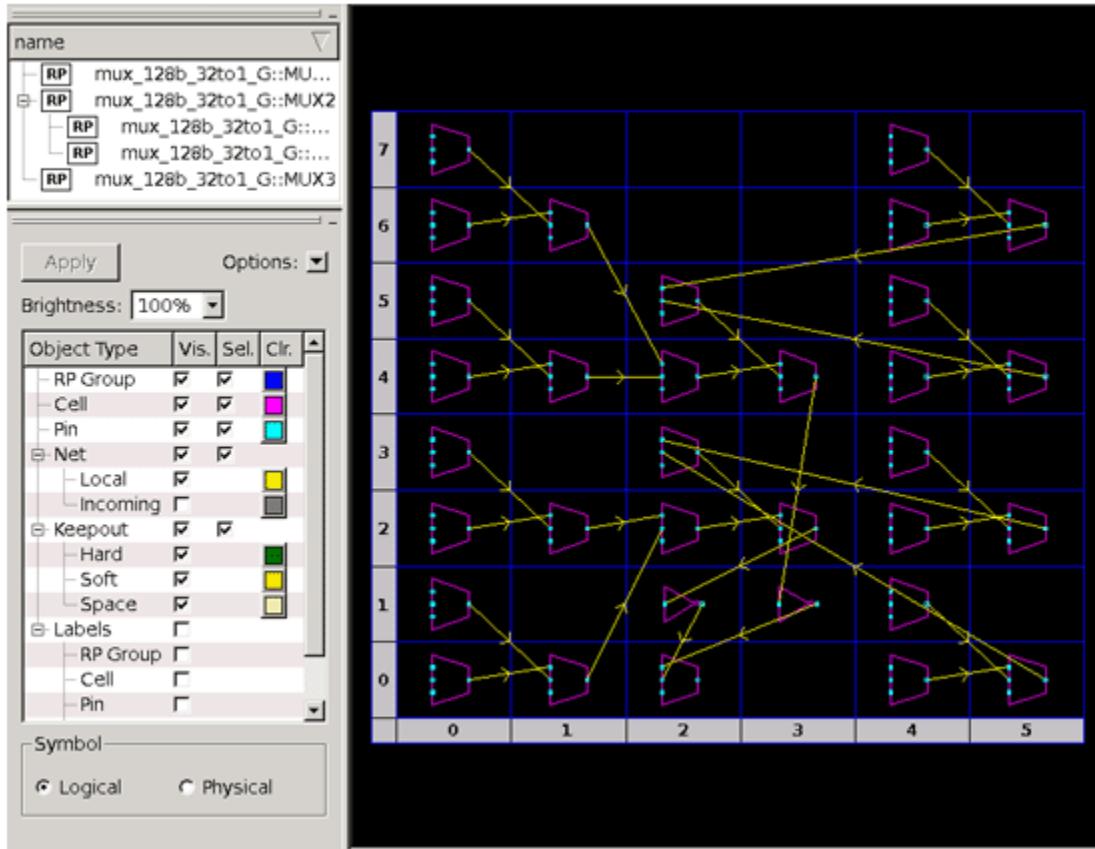


For more information about using the relative placement hierarchy view window, see IC Compiler Help.

Viewing Logical Connections in a Relative Placement Group

To view logical connections in a relative placement group, you can select a relative placement group and then open a relative placement view window by choosing View > New RP View of Selected RP from a relative placement window. [Figure 9-43](#) shows an example of a relative placement view window that displays relative placement cells in logical symbol view.

Figure 9-43 Relative Placement View Window



The relative placement view window displays one relative placement group and includes a relative placement group that contains any level of hierarchical relative placement groups. Objects inside the relative placement group are displayed with respect to their relative physical locations in grids. You have the options either to display cells in logical symbol view or to display them in the actual physical cell view. The relative placement view allows you to cross-select objects in other open views, such as the layout view.

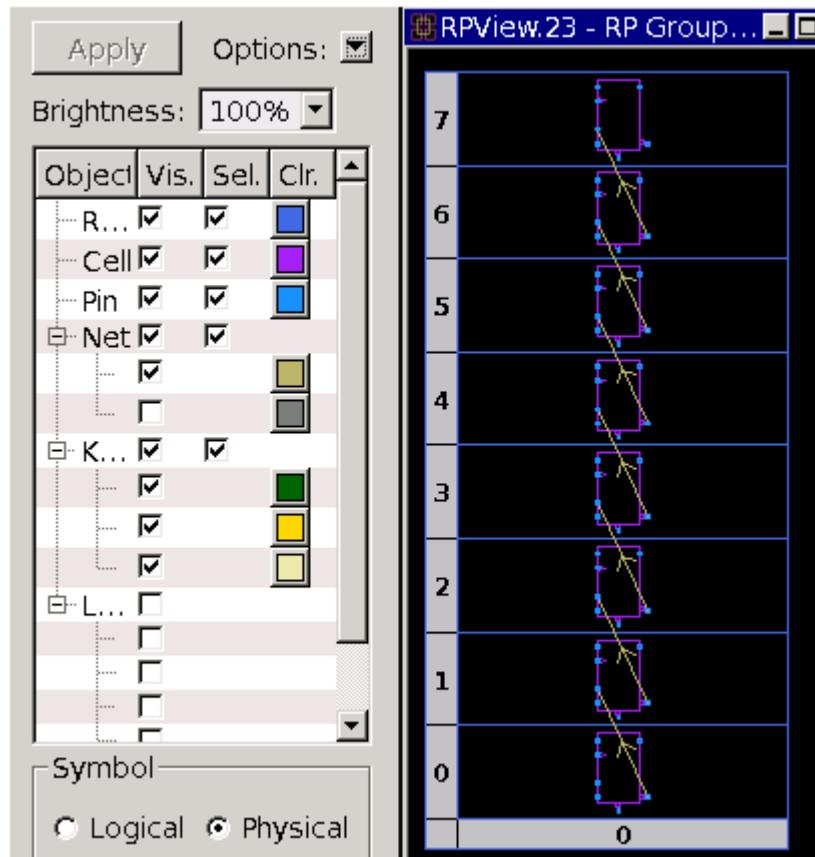
In the relative placement view window, you can perform the following tasks:

- Open a relative placement hierarchy view window.
- View cells in logical symbol or physical cell view.
- See net connectivity in grid view.
- Cross-select between all views.
- Zoom in or out of a hierarchical relative placement group.

- Expand or collapse a hierarchical relative placement group at any level.
- Edit relative placement groups.
- Change the structures of relative placement groups.

To view the settings of the selected relative placement group in a relative placement view window, choose View > Toolbar > View Settings in the GUI. The relative placement view setting toolbar, as shown in [Figure 9-44](#), lets you select objects, change the visibility, brightness, and color of objects, and display cells in either logical symbol view or physical cell view.

Figure 9-44 New Relative Placement View Setting Toolbar



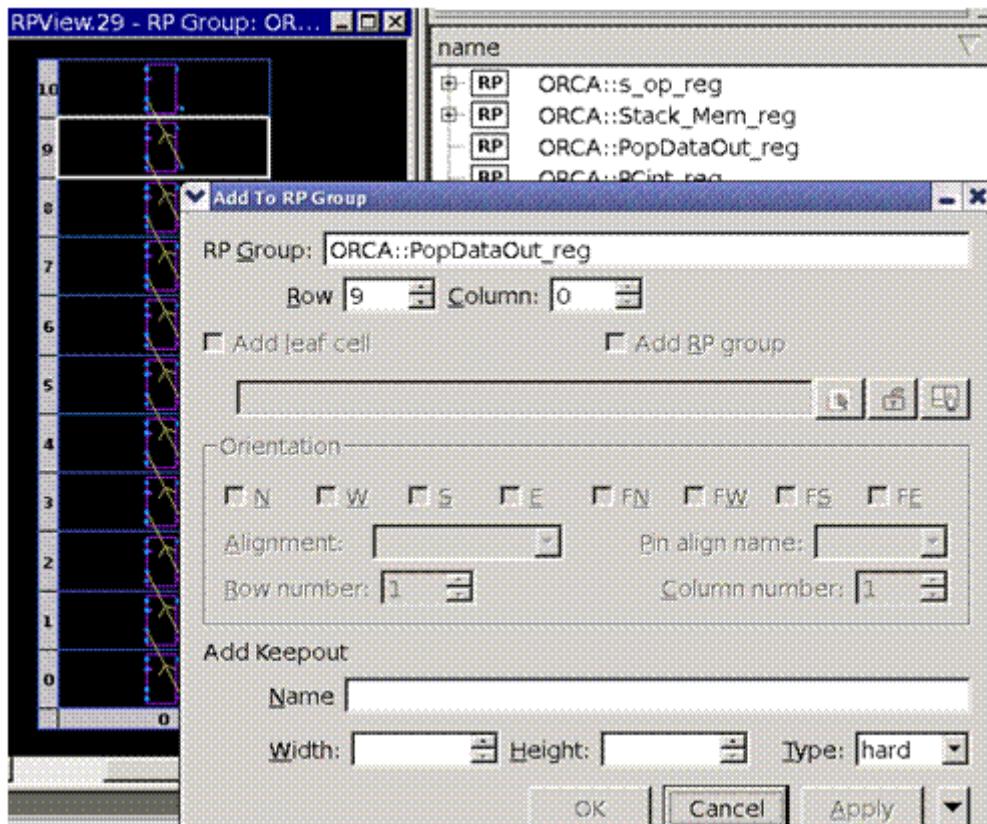
Editing Relative Placement Groups in a Relative Placement View Window

To open a relative placement view window, choose View > New RP View of Selected RP in a relative placement window. You can add, remove, drag, and switch relative placement objects in a relative placement view window.

To add relative placement groups, use the following steps:

1. Select the grid locations of the relative placement objects that you want to edit.
2. Right-click and choose Add, or click the  button from the toolbar.
The Add To RP Group dialog box appears, as shown in [Figure 9-45](#).
3. Add leaf cells, relative placement groups, or keepouts to that location.
4. Click OK or Apply.

Figure 9-45 Add to RP Group Dialog Box



To remove relative placement objects from the selected relative placement group, use the following steps:

1. Select relative placement objects to be removed.
2. Right-click to choose Remove, or click the  button from the toolbar.

To drag a relative placement object from one location to another, use the following steps:

1. Click the  button from the toolbar.
2. Select a relative placement object.

The selected object is highlighted in orange.

3. Drag the relative placement object to a new location.

If the new location contains an existing object, a warning dialog box appears.

To switch two relative placement objects between two locations, use the following steps:

1. Select two relative placement objects.
2. Click the  button from the toolbar.

To edit objects inside a hierarchical relative placement group, you need to display the hierarchical relative placement group in a relative placement view window. You can edit included relative placement groups but not instantiated relative placement groups in the hierarchical relative placement group. For more information about included and instantiated relative placement groups, see “[Adding Relative Placement Groups](#)” on page 9-35.

For more information about using the relative placement view window, see IC Compiler Help.

Editing the Structures of Relative Placement Groups

You can insert, remove, swap, and flip rows and columns to change the structures of existing relative placement groups in a relative placement view window. To open a relative placement view window, choose View > New RP View of Selected RP in a relative placement window. For more information about the relative placement window, see “[Using the Relative Placement Window](#)” on page 9-75.

To insert a row below a selected row,

1. Select a row where you want to insert a new row.
2. Choose Edit > Insert Row/Column Before.

A new row appears below the selected row.

To insert a column to the left of a selected column,

1. Select a column where you want to insert a new column.
2. Choose Edit > Insert Row/Column Before.

A new column appears to the left of the selected column.

To insert a row above a selected row,

1. Select a row where you want to insert a new row.
2. Choose Edit > Insert Row/Column After.

A new row appears above the selected row.

To insert a column to the right of a selected column,

1. Select a column where you want to insert a new column.
2. Choose Edit > Insert Row/Column After.

A new column appears to the right of the selected column.

To remove a row or column,

1. Select a row or column that you want to remove.
2. Choose Edit > Remove.

The selected row or column is removed from the relative placement group.

To swap two rows or columns,

1. Select two rows or columns that you want to swap.
2. Choose Edit > Switch.

The selected two rows or columns are swapped.

To flip rows or columns,

1. Select the rows or columns that you want to flip the objects.
2. Choose Edit > Flip Rows/Columns.

The objects inside the selected rows or columns are flipped.

Querying Relative Placement Groups and Objects

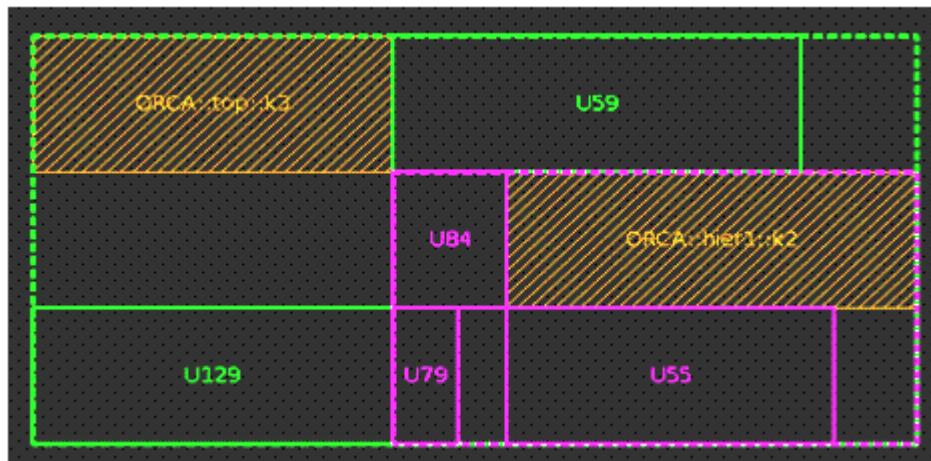
You can query relative placement groups and relative placement objects by using the relative placement attributes and query commands. The following sections describe how to query relative placement groups and objects.

- [Displaying Relative Placement Group Attributes](#)
- [Displaying Relative Placement Cell Attributes](#)
- [Displaying Relative Placement Keepout Attributes](#)
- [Reporting the Total Net Lengths of Relative Placement Groups](#)
- [Querying Relative Placement Groups](#)
- [Querying Objects in Relative Placement Groups](#)
- [Query Commands for Relative Placement Groups and Objects](#)

The examples in these sections use the relative placement group that contains a top-level block ORCA::top outlined in green and a lower level block ORCA::hier1 outlined in hot pink, as shown in [Figure 9-46](#).

The ORCA::top block consists of cell U59, cell U129, one keepout ORCA::top:k3, and one hierarchical relative placement group ORCA::hier1. The ORCA::hier1 block consists of cells U84, U79, U55, and one keepout ORCA::hier1:k2.

Figure 9-46 Relative Placement Group ORCA::top After Placement



You can use the following script to generate this relative placement group:

Example 9-13 Script for Creating the ORCA::top Relative Placement Group

```
create_rp_group hier1 -design ORCA -columns 2 -rows 2 \
    -utilization 1.000000
add_to_rp_group ORCA::hier1 -leaf U79 -column 0 -row 0
add_to_rp_group ORCA::hier1 -leaf U84 -column 0 -row 1
add_to_rp_group ORCA::hier1 -leaf U55 -column 1 -row 0
add_to_rp_group ORCA::hier1 -keepout k2 -column 1 -row 1 -width 25
create_rp_group top -design ORCA -columns 2 -rows 2 -utilization 1.000000
add_to_rp_group ORCA::top -leaf U129 -column 0 -row 0
add_to_rp_group ORCA::top -leaf U59 -column 1 -row 1
add_to_rp_group ORCA::top -keepout k3 -column 0 -row 1
add_to_rp_group ORCA::top -hierarchy ORCA::hier1 -column 1 -row 0
```

Displaying Relative Placement Group Attributes

To report one relative placement group attribute, use the `get_attribute` command. For example, to report the value of the `compress` attribute of the ORCA::top relative placement group, enter

```
icc_shell> get_attribute ORCA::top compress
```

To report all relative placement group attributes, use the `report_attribute -application` command. For example, the following command reports all the attributes of the ORCA::top relative placement group:

```
icc_shell> report_attribute -application ORCA::top
```

Design	Object	Type	Attribute Name	Value
ORCA	ORCA::top	string	alignment	bottom-left
ORCA	ORCA::top	string	bbox	Not placed
ORCA	ORCA::top	int	cell_id	8
ORCA	ORCA::top	int	columns	2
ORCA	ORCA::top	boolean	compress	false
ORCA	ORCA::top	string	cts_option	fixed_placement
ORCA	ORCA::top	string	design	ORCA
ORCA	ORCA::top	boolean	ignore	false
ORCA	ORCA::top	boolean	is_placed	false
ORCA	ORCA::top	boolean	is_top	true
ORCA	ORCA::top	string	move_effort	medium
ORCA	ORCA::top	string	name	ORCA::top
ORCA	ORCA::top	string	object_class	rp_group
ORCA	ORCA::top	int	object_id	953373
ORCA	ORCA::top	string	pin_align_name	Unknown
ORCA	ORCA::top	string	placed_orient	rp group is not placed
ORCA	ORCA::top	string	psynopt_option	size_only
ORCA	ORCA::top	string	route_opt_option	fixed_placement
ORCA	ORCA::top	int	rows	2
ORCA	ORCA::top	string	rp_height	18.450000
ORCA	ORCA::top	string	rp_width	20.910000

You can also display the attributes of a selected relative placement group in the GUI by right-clicking the relative placement group and choosing Properties.

Displaying Relative Placement Cell Attributes

To report one relative placement cell attribute, use the `get_attribute` command. For example, the following command reports the `rp_orientation` attribute of the U59 relative placement cell:

```
icc_shell> get_attribute U59 rp_orientation
```

To report all relative placement cell attributes, use the `report_attribute -application` command. For example, the following command reports all the attributes of the U59 relative placement cell:

```
icc_shell> report_attribute -application U59
```

Design	Object	Type	Attribute Name	Value
<hr/>				
...				
ORCA	U59	string	rp_alignment	bottom-right
ORCA	U59	int	rp_column	1
ORCA	U59	string	rp_group_name	ORCA::top
ORCA	U59	int	rp_num_columns	0
ORCA	U59	int	rp_num_rows	0
ORCA	U59	string	rp_orientation	S
ORCA	U59	string	rp_pin_align_name	Z
ORCA	U59	int	rp_row	1
...				
<hr/>				

Note that the attributes with an `rp_` prefix are a subset of the entire relative placement cell attributes. This subset, which describes the status of a cell in a relative placement group, exists only if the cell is contained in a relative placement group.

You can also display the attributes of a selected relative placement cell in the GUI by right-clicking the relative placement cell and choosing Properties.

To list all the attributes of relative placement cells in alphabetical order, use the `list_attributes` command. The command lists the following attributes: `rp_group_name`, `rp_row`, `rp_column`, `rp_num_rows`, `rp_num_columns`, `rp_alignment`, `rp_pin_align_name`, and `rp_orientation`.

Displaying Relative Placement Keepout Attributes

To report one relative placement keepout attribute, use the `get_attribute` command. For example, to report the `type` attribute of the ORCA::top::k3 relative placement keepout, enter

```
icc_shell> get_attribute ORCA::top::k3 type
```

To report all relative placement keepout attributes, use the `report_attribute -application` command. For example, the following command reports all the attributes of the ORCA::top::k3 relative placement keepout:

```
icc_shell> report_attribute -application ORCA::top::k3
```

Design	Object	Type	Attribute Name	Value
ORCA	ORCA::top::k3	string	bbox	Not placed
ORCA	ORCA::top::k3	int	cell_id	8
ORCA	ORCA::top::k3	string	height	Not specified
ORCA	ORCA::top::k3	string	name	ORCA::top::k3
ORCA	ORCA::top::k3	string	object_class	rp_group_keepout
ORCA	ORCA::top::k3	int	object_id	958721
ORCA	ORCA::top::k3	string	type	hard
ORCA	ORCA::top::k3	string	width	Not specified

You can also display the attributes of a selected relative placement keepout in the GUI by right-clicking the relative placement keepout and choosing Properties.

Reporting the Total Net Lengths of Relative Placement Groups

To report the total net lengths of relative placement groups, use the `get_rp_groups_net_length` command. The command returns a list of the total net lengths of each specified relative placement group. If your design is not routed, the command returns estimated values.

For example, the following command reports a list of total net lengths of individual relative placement groups, `design::rp1` and `design::rp2`:

```
icc_shell> get_rp_groups_net_length {design::rp1 design::rp2}
{100.001 1200.23}
```

Querying Relative Placement Groups

To query relative placement groups that contain specific objects or attribute values, use the `get_rp_groups` command with the following options:

- The `-of_objects` option

To find the relative placement group that contains leaf cell U129, enter

```
icc_shell> get_rp_groups -of_objects U129
{ORCA::top}
```

To find the relative placement group that contains the ORCA::hier1 hierarchical relative placement group, enter

```
icc_shell> get_rp_groups -of_objects ORCA::hier1
{ORCA::top}
```

To find the relative placement group that contains the ORCA::hier1::k2 relative placement keepout, enter

```
icc_shell> get_rp_groups -of_objects ORCA::hier1::k2
{ORCA::hier1}
```

- The `-filter` option

To find the relative placement group that is marked with the `is_top` attribute, enter

```
icc_shell> get_rp_groups -filter "is_top == TRUE"
{ORCA::top}
```

Querying Objects in Relative Placement Groups

You can use the following commands to find objects, including leaf cells, hierarchical relative placement groups, and keepouts that are in a relative placement group. Except the `rp_group_references` command, the other commands report relative placement objects in a single hierarchy level only.

- `rp_group_references`

To find all leaf cells that are directly included in the ORCA::top relative placement group, enter

```
icc_shell> rp_group_references -leaf ORCA::top
{U59 U129}
```

Alternatively, you can use the following command to find all leaf cells in the ORCA::top relative placement group:

```
icc_shell> get_cells -all -of_objects ORCA::top
{U59 U129}
```

To find all leaf cells in the ORCA::top relative placement group and down the hierarchy, enter

```
icc_shell> rp_group_references -all_leaf ORCA::top
{U59 U129 U55 U79 U84}
```

- `rp_group_inclusions`

To find all hierarchical relative placement groups in the ORCA::top relative placement group, enter

```
icc_shell> rp_group_inclusions ORCA::top
{ORCA::hier1}
```

- `get_rp_group_keepouts -of_objects`

To find all relative placement keepouts in the ORCA::top relative placement group, enter

```
icc_shell> get_rp_group_keepouts -of_objects ORCA::top
{ORCA::top::k3}
```

To find relative placement objects that have specific attribute values, use the following commands with the `-filter` option:

- `get_cells -filter`

To find the leaf cells that occupy column position 1 in all relative placement groups, enter

```
icc_shell> get_cells -filter {rp_column==1} -hierarchical
{U59 U55}
```

- `get_rp_group_keepouts -filter`

To find the relative placement keepouts whose width equals 25, enter

```
icc_shell> get_rp_group_keepouts -filter "width==25"
{ORCA::hier1::k2}
```

To find the relative placement object at a specific location or vice versa, use the following commands:

- `get_cells -filter`

To find the leaf cell at position 0 0 (column 0, row 0) in the ORCA::top relative placement group, enter

```
icc_shell> get_cells \
      -filter {rp_group_name=="ORCA::top" && rp_column==0 && rp_row==0}
{U129}
```

If the leaf cell is not in the current hierarchy, specify the `-hierarchical` option.

- `get_location`

To get the grid location of the U129 relative placement cell in the ORCA::top relative placement group, enter

```
icc_shell> get_location U129 -rp_group ORCA::top
0 0
```

For more information about getting the grid location of a relative placement cell or group, see “[Locating a Relative Placement Group](#)” on page 9-68.

Query Commands for Relative Placement Groups and Objects

[Table 9-4](#) lists the commands available for querying particular types of relative placement groups to annotate, edit, and view.

Table 9-4 Commands for Querying Relative Placement Groups

To return collections for this	Use this command
Relative placement groups that match certain criteria	<code>get_rp_groups</code>
Total net lengths of individual relative placement groups	<code>get_rp_groups_net_length</code>
All or specified relative placement groups and the included and instantiated groups they contain in their hierarchies	<code>all_rp_groups</code>
All or specified relative placement groups that contain included or instantiated groups	<code>all_rp_hierarchicals</code>
All or specified relative placement groups that contain included groups	<code>all_rp_inclusions</code>
All or specified relative placement groups that contain instantiated groups	<code>all_rp_instantiations</code>
All or specified relative placement groups that directly embed leaf cells (added to a group by <code>add_to_rp_group -leaf</code>) or instantiated groups (added to a group by <code>add_to_rp_group -hierarchy -instance</code>) in all or specified relative placement groups in a specified design or the current design	<code>all_rp_references</code>
Directly embedded included groups (added to a group by <code>add_to_rp_group -hierarchy</code>) in all or specified groups	<code>rp_group_inclusions</code>
Directly embedded instantiated groups (added to a group by <code>add_to_rp_group -hierarchy -instance</code>) in all or specified groups	<code>rp_group_instantiations</code>

Table 9-4 Commands for Querying Relative Placement Groups (Continued)

To return collections for this	Use this command
Directly embedded leaf cells (added to a group by <code>add_to_rp_group -leaf</code>), directly embedded included cells that contain hierarchically instantiated cells (added to the included group by <code>add_to_rp_group -hierarchy -instance</code>), or both in all or specified relative placement groups	<code>rp_group_references</code>
Create a collection of the specified relative placement group keepouts	<code>get_rp_group_keepouts</code>

Saving Relative Placement Information

The relative placement information is automatically saved in the Milkyway design database when you save the design in Milkyway format using the `save_mw_cel` command.

You can also save the relative placement information to a file, creating a Tcl script for re-creating relative placement groups and their objects on the same design. To do this, use the `write_rp_groups -output script` command. You must either specify which relative placement groups to write commands for or specify the `-all` option to write commands for all relative placement groups.

For example, to save all the relative placement groups to disk, remove the information from the design, and then re-create the information on the design, enter the following commands:

```
icc_shell> get_rp_groups
{mul::grp_mul ripple::grp_ripple example3::top_group}
icc_shell> write_rp_groups -all -output my_groups.tcl
1
icc_shell> remove_rp_groups -all -quiet
1
icc_shell> get_rp_groups
Error: Can't find objects matching '*'. (UID-109)
icc_shell> source my_groups.tcl
{example3::top_group}
icc_shell> get_rp_groups
{example3::top_group ripple::grp_ripple mul::grp_mul}
```

Alternatively, you can save the relative placement information from the relative placement view window by choosing RP > Write RP Groups. For more information about opening a relative placement view window, see “[Using the Relative Placement Window](#)” on page 9-75.

By default, the `write_rp_groups` command writes out commands for creating the specified relative placement groups and to add leaf cells, hierarchical groups, and keepouts to these

groups. The commands for generating subgroups within hierarchical groups are not written. You can modify the default behavior by using the options described in [Table 9-5](#).

To remove all relative placement information from relative placement groups, use the `remove_rp_group` command.

Table 9-5 write_rp_groups Options

To do this	Use this option
Write all the relative placement groups within the hierarchy of the relative placement groups. If you omit this option, subgroups are not written.	-hierarchy
Write only <code>create_rp_group</code> commands to the script.	-create
Write only <code>add_to_rp_group -leaf</code> commands to the script.	-leaf
Write only <code>add_to_rp_group -keepout</code> commands to the script.	-keepout
Write only <code>add_to_rp_group -hierarchy -instance</code> commands to the script.	-instance
Write only <code>add_to_rp_group -hierarchy</code> commands to the script.	-include

Ignoring Relative Placement Constraints

You can direct the tool to ignore relative placement constraints annotated to the design. An example of this is when you want to confirm that relative placement is helpful to the QoR. To skip structured placement for relative placement groups that you specify, use the `-ignore` option with either the `create_rp_group` or `set_rp_group_options` command.

When you direct the tool to ignore relative placement constraints, it places the parts of the relative placement groups as if the group had no relative placement information.

To remove the `ignore` attribute and cause the tool to consider the relative placement constraints, use the `remove_rp_group_options -ignore` command.

Removing Relative Placement Groups

You can remove relative placement groups in the relative placement hierarchy browser, relative placement editing dialog box, relative placement view window, or from the command line.

To remove relative placement groups in the relative placement hierarchy browser,

1. Select the groups you want to remove in the left pane.
2. Right-click and choose Delete Selected RP Groups from the pop-up menu.

For more information about the relative placement hierarchy browser, see “[Using the Relative Placement Hierarchy Browser](#)” on page 9-69.

To remove relative placement groups in the relative placement group editing dialog box, choose Placement > Edit RP Group in the GUI. For more information, see “[Editing and Creating Relative Placement Groups](#)” on page 9-73.

To remove relative placement groups in the relative placement view window, see “[Editing Relative Placement Groups in a Relative Placement View Window](#)” on page 9-80 for more information.

To remove relative placement groups from the command line, run the `remove_rp_groups` command. You can remove all relative placement groups (by specifying the `-all` option), or you can specify which relative placement groups to remove. If you specify a list of relative placement groups, only the specified groups (and not groups included or instantiated within the specified group) are removed. To remove the included and instantiated groups of the specified groups, you must specify the `-hierarchy` option.

Note:

When you remove a relative placement group, the memory it occupies is freed to be reused by this same process. However, memory is not returned to the operating system until you exit `icc_shell`.

For example, to remove the relative placement group named `grp_ripple` and confirm its removal, enter

```
icc_shell> get_rp_groups
{mul::grp_mul ripple::grp_ripple example3::top_group}
icc_shell> remove_rp_groups ripple::grp_ripple
Removing rp group 'ripple::grp_ripple'
1
icc_shell> get_rp_groups *grp_ripple
Error: Can't find object 'grp_ripple'. (UID-109)
icc_shell> remove_rp_groups -all
Removing rp group 'mul::grp_mul'
Removing rp group 'example3::top_group'
```

Changing Relative Placement Information

You can change a relative placement group. For example, you can

- Change the attributes of a relative placement group (anchor location, alignment, utilization, preservation during clock tree synthesis, and preservation during routing)
- Rename a group
- Remove objects from a group
- Change objects within a group
- Change the attributes of cells within a group
- Change the dimensions of keepouts within a group

Reporting the Attributes of a Relative Placement Group

To report the attributes of one or more relative placement groups (as set by the `create_rp_group` or `set_rp_group_options` command), use the `report_rp_group_options` command.

```
icc_shell> report_rp_group_options rp_groups
```

You can also use the `report_attribute` command to report the attributes of all relative placement groups in your design.

```
icc_shell> report_attribute -application -class rp_group rp_group
```

In addition to the attributes reported by the `report_rp_group_options` command, the `report_attribute` command also lists the number of rows and columns of each relative placement group.

Changing the Options of a Relative Placement Group

When you create a relative placement group by using the `create_rp_group` command, you set the attributes for the relative placement group. To change the attributes, you use one of the following two methods:

- Set the attribute values.
- Reset the attributes to their defaults by using the `remove_rp_group_options` command.

Setting Relative Placement Group Attributes

You can set relative placement group attributes either in the relative placement hierarchy browser or from the command line.

To set relative placement group attributes in the relative placement hierarchy browser,

1. Select the groups you want to modify in either the left or right pane (relative placement groups are identified by the  icon).
2. Right-click and choose “Set RP Options.”
The “Set RP Options” dialog box appears.
3. Update the attributes, as described in [Table 9-6 on page 9-95](#).

For more information about the relative placement hierarchy browser, see [“Using the Relative Placement Hierarchy Browser” on page 9-69](#).

To set relative placement group attributes from the command line, run the `set_rp_group_options` command. You must specify the group name and at least one option; otherwise, this command has no effect.

For example, to specify that all cells in the ORCA::top relative placement group can be compressed during placement, enter

```
icc_shell> set_rp_group_options ORCA::top -placement_type compression
```

To skip optimization for leaf cells in the ORCA::top relative placement group during the `psynopt` command, enter

```
icc_shell> set_rp_group_options ORCA::top -psynopt_option fixed_placement
```

To set the utilization to 80 percent for the block1::misc1 group, enter

```
icc_shell> set_rp_group_options block1::misc1 -utilization .80  
{block1::misc1}
```

The command returns a collection of relative placement groups for which attributes have been changed. If no attributes change for any object, an empty string is returned.

Table 9-6 describes the relative placement group attributes and the Set RP Options dialog box fields or command options used to change them.

Table 9-6 Relative Placement Group Attributes

Attribute	Dialog box fields	Command line options
<p>Specify an alignment method. See “Aligning Leaf Cells Within a Column” on page 9-23.</p>	Alignment	-alignment
<p>Allow nonrelative placement cells in relative placement groups. See “Exposing Unused Space in Relative Placement Groups” on page 9-62.</p>	Allow non rp cells	-allow_non_rp_cells
<p>Specify a bottom-left or bottom-right corner for the anchored relative placement group. See “Specifying a Corner for an Anchored Relative Placement Group” on page 9-15.</p>	Anchor corner	-anchor_corner
<p>Prevent inserting buffers inside the areas of placed relative placement groups. See “Buffering Strategy for Relative Placement Groups” on page 9-64.</p>	Auto blockage	-auto_blockage
<p>Specify if the tool can automatically orient the cells in the specified relative placement group. See “Orientation of Relative Placement Groups” on page 9-30.</p>	Cell Orient Opt	-cell_orient_opt
<p>Specify clock tree synthesis preservation. You can specify <code>fixed_placement</code> or <code>size_only</code>. See “Postplacement Optimization of Relative Placement Groups” on page 9-62.</p>	CTS option	-cts_option
<p>Prevent inserting buffers into nets inside relative placement groups. See “Buffering Strategy for Relative Placement Groups” on page 9-64.</p>	Disable buffering	-disable_buffering

Table 9-6 Relative Placement Group Attributes (Continued)

Attribute	Dialog box fields	Command line options
Specify the orientation of relative placement groups. You can specify default, N, FN, S, and FS. See “Orientation of Relative Placement Groups” on page 9-30.	Group orientation	-group_orient
Ignore relative placement constraints. See “Ignoring Relative Placement Constraints” on page 9-91.	Ignore placement	-ignore
Legalize relative placement groups. You can specify low, medium, or high. See “Controlling Movement When Legalizing Relative Placement Groups” on page 9-59.	Effort	-move_effort
Specify the group alignment pin. See “Aligning Leaf Cells Within a Column” on page 9-23.	Pin alignment	-pin_align_name
Specify how to treat fixed cells in the floorplan during legalization. See “Handling Fixed Cells During Relative Placement” on page 9-53.	Place around fixed cells	-place_around_fixed_cells
Specify a placement type for relative placement groups. You can specify bit_slice, vertical_compression, and compression. See “Applying Compression to Relative Placement Groups” on page 9-18.	Placement type	-placement_type
Apply compression in the horizontal direction to a relative placement group. See “Applying Compression to Relative Placement Groups” on page 9-18.	Compress	-placement_type compression

Table 9-6 Relative Placement Group Attributes (Continued)

Attribute	Dialog box fields	Command line options
Specify the physical optimization preservation attribute. You can specify <code>fixed_placement</code> , <code>size_only</code> , or <code>all_optimization</code> . See “ Postplacement Optimization of Relative Placement Groups ” on page 9-62.	Psynopt option route_opt option	<code>-psynopt_option</code> <code>-route_opt_option</code>
Specify the routing preservation attribute. You can specify <code>fixed_placement</code> or <code>in_place_size_only</code> . See “ Postplacement Optimization of Relative Placement Groups ” on page 9-62.	route_opt option	<code>-route_opt_option</code>
Allow keepouts over tap cells. See “ Adding Keepouts ” on page 9-49.	Allow keepout over tapcell	<code>-allow_keepout_over_tapcell</code>
Specify the utilization percentage.	Utilization	<code>-utilization</code>
Specify the anchor location. See “ Anchoring Relative Placement Groups ” on page 9-13.	X offset Y offset	<code>-x_offset</code> <code>-y_offset</code>

Removing Relative Placement Group Attributes

You can remove relative placement group attributes either in the relative placement hierarchy browser or from the command line.

To remove relative placement group attributes in the relative placement hierarchy browser (Placement > New RP Hierarchy View),

1. Select the groups you want to modify in either the left or right pane (relative placement groups are identified by the  icon).
2. Right-click and choose “Set RP Options.”
The “Set RP Options” dialog box appears.
3. Remove the attributes by setting them to empty values.

To remove relative placement group attributes from the command line, run the `remove_rp_group_options` command. You must specify the group name and at least one option; otherwise, this command has no effect.

For example, to remove the `x_offset` attribute for the `block1::misc1` relative placement group, enter

```
icc_shell> remove_rp_group_options block1::misc1 -x_offset
{block1::misc1}
```

The command returns a collection of relative placement groups for which attributes have been changed. If no attributes change for any object, an empty string is returned.

[Table 9-7](#) describes changes you can make with the `remove_rp_group_options` command.

Table 9-7 remove_rp_group_options Options

To do this	Use this option
Reset the anchor location. Removing an offset attribute resets that coordinate to 0. See “ Anchoring Relative Placement Groups ” on page 9-13.	-x_offset -y_offset
Remove the ignore setting for this relative placement group. See “ Ignoring Relative Placement Constraints ” on page 9-91.	-ignore
Remove the automatic cell orientation setting in the specified relative placement group. See “ Orientation of Relative Placement Groups ” on page 9-30.	-cell_orient_opt
Remove the buffering setting inside relative placement groups during the <code>psynopt</code> and <code>route_opt</code> steps. See “ Buffering Strategy for Relative Placement Groups ” on page 9-64.	-auto_blockage
Remove the buffering setting into nets inside relative placement groups during the <code>psynopt</code> and <code>route_opt</code> steps. See “ Buffering Strategy for Relative Placement Groups ” on page 9-64.	-disable_buffering

Renaming a Group

You cannot rename a group directly. If you need to rename a group, you must remove it, and then create a new group that duplicates the removed group but has the new name.

Removing Objects From a Group

You can remove objects from a relative placement group in the relative placement hierarchy browser, in the RP Editing dialog box, or from the command line.

To remove objects from a relative placement group in the relative placement hierarchy browser (Placement > New RP Hierarchy View),

1. Select the objects you want to remove in the right pane (relative placement groups are identified by the  icon, leaf cells are identified by the  icon, keepouts are identified by the  icon, and locations containing both a leaf cell and a keepout are identified by the  icon).
2. Invoke the pop-up menu by right-clicking.
3. Right-click and choose “Remove Selected Cells and RP Groups” or “Remove Selected Keepouts”.

For more information about the relative placement hierarchy browser, see [“Using the Relative Placement Hierarchy Browser” on page 9-69](#).

To remove objects from a relative placement group from the command line, use the `remove_from_rp_group` command. You can remove leaf cells (`-leaf`), included groups (`-hierarchy`), instantiated groups (`-hierarchy -instance`), and keepouts (`-keepout`).

When you remove objects from a group, the space previously occupied by the removed objects is left unoccupied.

For example, to remove leaf cell `carry_in_1` from `grp_ripple`, enter

```
icc_shell> remove_from_rp_group ripple::grp_ripple -leaf carry_in_1
```

To remove objects from a relative placement group in the RP Editing dialog box, see [“Editing and Creating Relative Placement Groups” on page 9-73](#).

Changing Specific Objects Within a Group

You can change a specific cell within a group. For example, you can move a cell from one position to another or change its alignment pin name or orientation.

To change an object within a group,

1. Remove the object from the group by using the `remove_from_rp_group` command.
2. Reinsert the object by using the appropriate `add_to_rp_group` command.

Changing the Attributes of Relative Placement Cells

You specify the following constraints for a relative placement group on a per-cell basis: column, row, alignment, pin alignment, orientation, row position, and column position. You use the `add_to_rp_group` command with the appropriate options to set these constraints. Among these constraints, you can modify the alignment, pin alignment, and orientation constraints by changing the `rp_alignment`, `rp_pin_align_name`, and `rp_orientation` relative placement cell attributes respectively. You cannot change the other attributes after they are created. To change them, you must use the `remove_from_rp_group` command followed by the `add_to_rp_group` command.

To change the `rp_alignment`, `rp_pin_align_name`, and `rp_orientation` attributes of one or more relative placement cells, use the `set_attribute` command. For example, to set a flipped-north (FN) orientation for the U98 relative placement cell, enter

```
icc_shell> set_attribute U98 rp_orientation FN
```

You can remove the `rp_alignment`, `rp_pin_align_name`, and `rp_orientation` attributes from relative placement cells by using the `remove_attribute` command. For example, to remove the `rp_orientation` attribute from the U59 relative placement cell, enter

```
icc_shell> remove_attribute U59 rp_orientation
```

Changing the Dimensions of Relative Placement Keepouts

To change the width and height of a relative placement keepout, specify the `width` and `height` attributes with the `set_attribute` command.

For example, the following command changes the height of all relative placement keepouts to 2 site heights:

```
icc_shell> set_attribute [get_rp_group_keepouts *] height 2
```

Deriving Relative Placement Groups

You can derive relative placement groups from your design by either specifying the cells contained in the group or specifying the physical location of the group.

Note:

The IC Compiler tool does not derive hierarchical relative placement groups. Derived relative placement groups are always flat.

You can use this capability to migrate and reuse structured netlists.

To derive relative placement groups from your design,

- Use the `extract_rp_group` command, which generates a script containing `create_rp_group` commands for the extracted groups.
- Use the `order_rp_groups` command to order the extracted groups to create a linear placement of groups.

This capability does not generate all possible relative placement groups but can serve as a starting point in many cases. You can refine the resulting groups by editing the script generated by `extract_rp_group`.

Extracting a Relative Placement Group

The `extract_rp_group` command helps you create relative placement structures at an abstract level. You can extract a group by either specifying the cells in the group with the `-objects` option or specifying a bounding box from which to extract the group with the `-coordinates` option. Alternatively, you can use the relative placement editing dialog box to extract a relative placement group. For more information, see “[Editing and Creating Relative Placement Groups](#)” on page 9-73.

The `extract_rp_group` command annotates the extracted relative placement group to the design and generates the `create_rp_group` and `add_to_rp_group` commands to define the relative placement group. By default, these commands are written to the screen. To write the commands to a script file, use the `-output` option. To append the commands to an existing script file, specify the `-append` option in addition to the `-output` option. You can refine the extracted group by editing the script file.

By default, IC Compiler creates one column and as many rows as the number of cells that are specified. To change the default behavior, specify the desired number of rows with the `-rows` option and columns with the `-columns` option. The specified cells are added from left to right starting from the bottom row according to the cell names alphabetically.

For example, to create a relative placement group for a bank of registers whose names start with `Crnt_Instrn_1_reg`, annotate the group to the design, and save the script to a file called `rp.tcl`, enter

```
icc_shell> extract_rp_group \
    -group_name Crnt_Instrn_1_reg \
    -objects [get_cells -hierarchical Crnt_Instrn_1_reg*] \
    -output rp.tcl
```

To create a relative placement group containing the cells within the bounding box with coordinates (0,0) and (100,100) and annotate the group to the design, enter

```
icc_shell> extract_rp_group -group_name ph1 \
    -physical -coordinates { 0 0 100 100 }
```

Alternatively, you can use the GUI to extract a relative placement group by choosing Placement > Extract RP Group.

Ordering Extracted Relative Placement Groups

You can use the `order_rp_groups` command to create a linear placement of the specified top-level relative placement groups.

The `order_rp_groups` command generates the `create_rp_group` and `add_to_rp_group` commands to define the hierarchical relative placement group. The generated relative placement group has one row and as many columns as the number of the hierarchical relative placement groups that are specified. By default, these commands are written to the screen. To write the commands to a script file, use the `-output` option. To append the commands to an existing script file, specify the `-append` option in addition to the `-output` option. You can refine the group by editing the script file.

For example, to define a hierarchical relative placement group for a related set of register banks, enter

```
icc_shell> extract_rp_group \
    -group_name Crnt_Instrn_1_reg \
    -objects [get_cells -hierarchical Crnt_Instrn_1_reg*] \
    -output Crnt_Instrn.tcl
icc_shell> extract_rp_group \
    -group_name Crnt_Instrn_2_reg \
    -objects [get_cells -hierarchical Crnt_Instrn_2_reg*] \
    -output Crnt_Instrn.tcl -append
icc_shell> order_rp_groups \
    -group_name Crnt_Instrn_Reg \
    {ORCA::Crnt_Instrn_1_Reg ORCA::Crnt_Instrn_2_Reg} \
    -output Crnt_Instrn.tcl -append
```

Summary of Relative Placement Commands

[Table 9-8](#) shows some of the key commands used to perform relative placement.

Table 9-8 Relative Placement Commands

Command	Described in section
create_rp_group	See “ Creating Relative Placement Groups ” on page 9-10.
add_to_rp_group	See “ Adding Objects to a Group ” on page 9-21.
modify_rp_groups	See “ Changing the Structures of Relative Placement Groups ” on page 9-41.
legalize_rp_placement	See “ Legalizing Relative Placement Groups in a Placed Design ” on page 9-60.
check_rp_groups	See “ Checking Relative Placement Constraints ” on page 9-65.
write_rp_groups	See “ Saving Relative Placement Information ” on page 9-90.
remove_rp_groups	See “ Removing Relative Placement Groups ” on page 9-92.
report_rp_group_options	See “ Reporting the Attributes of a Relative Placement Group ” on page 9-93.
set_rp_group_options	See “ Changing the Options of a Relative Placement Group ” on page 9-93.
remove_rp_group_options	See “ Removing Relative Placement Group Attributes ” on page 9-97.
remove_from_rp_group	See “ Removing Objects From a Group ” on page 9-99.
extract_rp_group	See “ Extracting a Relative Placement Group ” on page 9-101.
order_rp_groups	See “ Ordering Extracted Relative Placement Groups ” on page 9-102.

Limitations of Relative Placement

Relative placement has these limitations:

- When IC Compiler estimates that the size of a relative placement group is not suitable for the given floorplan, IC Compiler might fail to maintain the relative placement constraints during placement. To maintain relative placement information precisely, there must be enough space for relative placement groups without overlapping placement obstructions in the design floorplan.
- If your design contains multiheight cells, relative placement constraints might not be honored, because the legalizer might move relative placement cells to a valid site and leave an unintended gap.
- There is no limit on the number of cells in a relative placement group. However, if your design has many relative placement groups, coarse placement returns overlapping group locations at times, resulting in misalignment. In such cases, a warning appears after coarse placement.

10

Using Hierarchical Models

The following topics describe how to create hierarchical models such as block abstractions and use them in the IC Compiler optimization flows:

- [Overview of Block Abstractions](#)
- [Creating Block Abstractions](#)
- [Using Block Abstractions](#)
- [Performing Top-Level Optimization](#)
- [Performing Transparent Interface Optimization](#)

For information about extracted timing models, see the Extracted Timing Models chapter in the *PrimeTime User Guide*.

Overview of Block Abstractions

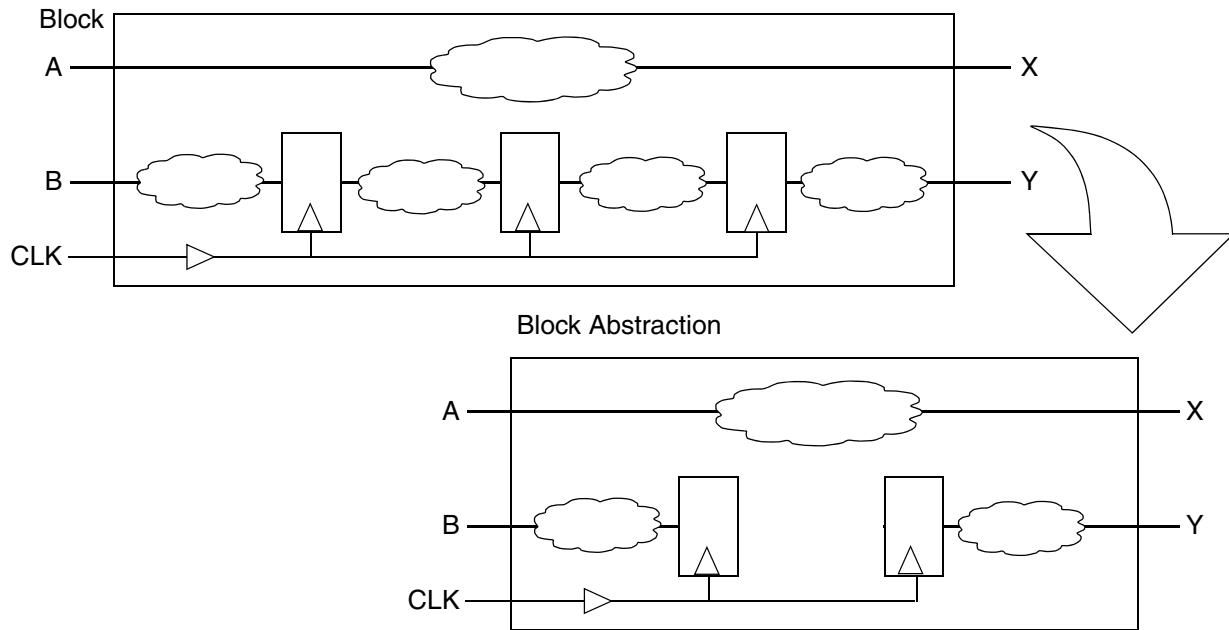
In a block abstraction, the gate-level netlist for the block is modeled by a partial gate-level netlist that contains only the required interface logic of the block and possibly the logic that you manually associate with the interface logic. All other logic is removed.

[Figure 10-1](#) shows a block and its block abstraction, where the logic is preserved between

- The input port and the first register of each timing path
- The last register of each timing path and the output port

Logic associated with pure combinational input-port-to-output-port timing paths (A to X) is also preserved. Clock connections to the preserved registers are kept as well. The register-to-register logic is discarded.

Figure 10-1 A Block and Its Block Abstraction



Supported Flows With Block Abstractions

The IC Compiler tool supports using block abstractions in a variety of flows, including those with

- Channeled or abutted layout styles

Channels can exist between blocks in a channeled layout style, but not in an abutted layout style.

- IEEE 1801 Unified Power Format (UPF)

The tool retains the following UPF objects in a block abstraction:

- All power-switch cell instances, irrespective of whether they are part of the interface logic or not, for proper supply net connectivity and always-on synthesis
- All retention registers, irrespective of whether they are part of the interface logic or not
- Level-shifter cells and isolation cells that are part of the interface logic
- Control logic for the retained power-switch cells, retention registers, isolation cells, and level-shifter cells

When the tool removes a UPF netlist object, it either removes or updates the associated UPF constraint. The tool ensures that the UPF strategies stored in the block abstraction are consistent with the UPF netlist objects.

When the tool links the top-level netlist, it automatically propagates the UPF objects in the block abstraction to the top level.

- Multicorner-multimode

The tool automatically detects the presence of multiple corners and multiple modes and retains the interface logic involved in the interface timing paths of each scenario.

- Hierarchical signal integrity

The tool can create block abstractions with coupling capacitance and signal integrity information for use with top-level signal integrity analysis and optimization. In this flow, the block abstractions are shielded at the top level to prevent crosstalk between the nets within the block abstraction and those at the top level.

Creating Block Abstractions

To create a block abstraction for the current design, use the `create_block_abstraction` command. When you create a block abstraction, the interface logic in the design's CEL view is annotated on the design.

When a block abstraction is created, the `create_block_abstraction` command

- Determines the interface logic

The following netlist objects are part of interface logic:

- All cells, pins, and nets in timing paths from input ports to registers or output ports
- All cells, pins, and nets in timing paths to output ports from registers or input ports
- Any logic in the connection from a master clock to generated clocks
- The clock trees that drive interface registers, including any logic in the clock tree
- The longest and shortest clock paths from the clock ports
- The side-load cells of all non-ideal and non-DRC disabled nets

- Ignores an input or inout port if the percentage of the total registers in the transitive fanout of the port is greater than or equal to the threshold percentage specified by the `abstraction_ignore_percentage` variable.

The following netlist objects are retained for ports that are ignored:

- Connections to the already identified interface logic of the other ports
- Minimum and maximum critical timing paths
- Ignores any case analysis settings. Any case analysis settings that are required must be specified at the top level by using the `set_case_analysis` command
- Assumes all latches found in interface logic are potential borrowers; thus, all logic from I/O ports to flip-flops or output ports are identified as belonging to interface logic
- Extracts and stores detailed parasitics as part of the abstraction information for each specified TLUPlus file and temperature
- Signal integrity (SI) and coupling capacitance information is retained when the design is track assigned or detail routed
- Calculates power consumption of the design and stores that information as attributes on the design. This information is used by the `report_power` command during the final assembly step of the entire chip

To include additional objects in the block abstraction, use the `-include` option with the `create_block_abstraction` command.

After you have created a block abstraction, you must use the `save_mw_cel` command to save it.

To identify blocks that have corresponding block abstractions in the Milkyway design library, use the following commands:

```
icc_shell> set BLOCKS_LIST {block1 block2 ...}  
icc_shell> get_mw_cels -filter "@has_block_abstraction == true" \  
$BLOCKS_LIST
```

Creating a Block Abstraction for Multicorner-Multimode Usage

The `create_block_abstraction` command automatically detects the presence of multiple scenarios (multiple modes or corners) and determines the interface logic for each scenario. The interface logic identified for each scenario is retained as the interface logic of the block abstraction. If an interface timing path is disabled in one scenario but enabled in another, the path is included in the interface logic.

Using Block Abstractions

A block abstraction is a structural interface logic model that enables optimization of the interface logic within the blocks during top-level optimization. This increases the scope for postroute optimization at the top level, and is useful when top-level optimization alone cannot achieve timing closure.

The following topics describe how to use block abstractions in a top-level design:

- [Linking to Block Abstractions](#)
- [Reporting and Querying Block Abstractions](#)
- [Checking the Readiness of Block Abstractions](#)
- [Applying Top-Level Constraints](#)
- [Viewing Block Abstractions in the GUI](#)
- [Using Block Abstractions in a Multicorner-Multimode Flow](#)
- [Handling Nested Physical Hierarchy](#)

Linking to Block Abstractions

To specify a list of block references to be linked with block abstractions, use the `set_top_implementation_options -block_references` command. To link to block abstractions, a block's CEL view should have the same name as the design name.

If an ETM and a block abstraction exists for a block, the tool links to

- The ETM, if it is included as a link library.
- The block abstraction, if the ETM is not included as a link library and the block abstraction has been specified by using the `set_top_implementation_options -block_references` command.

During linking, the tool loads only the block abstraction information annotated in the CEL view to the top level, and does not load the complete CEL view.

To verify the options that are set by using the `set_top_implementation_options` command, use the `report_top_implementation_options` command. The command reports the options in the following format:

Block Reference	Optimize Interface	Optimize Shared Logic	Size Only Mode	Host Options
block1	true	false	off	my_hosts
block2	true	false	off	my_hosts

The tool does not propagate any `dont_touch` attributes you apply at the block level to the top level. After you link the top-level design, reapply the `dont_touch` attributes on the objects within the block by using the `set_dont_touch` command from the top level.

Reporting and Querying Block Abstractions

To report on block abstractions that are linked to the current design, use the `report_block_abstraction` command, which reports the following information:

- Block reference name
- Block instance name
- Block version
- Full path of the library from which the block was loaded
- Date and time of last modification of the block
- Cell count and compression statistics for each of the blocks

- Cell count and compression statistics for the top design with blocks
- TLUPlus settings
- Parasitics data of the block
- User-defined multicorner-multimode scenario mappings between the top level and blocks
- UPF information

To create a collection of cells that are within block abstractions, use the following command at the top level:

```
icc_shell> get_cells -hierarchical * \
-filter "is_block_abstraction == true"
```

Checking the Readiness of Block Abstractions

To check the readiness of the block abstractions for placement optimization, clock tree synthesis, or routing optimization, use the `check_block_abstraction` command

The `check_block_abstraction` command checks if the blocks linked to the top-level design are ready for placement, clock, and route optimizations. If you do not specify the `-stage` option, the command performs the default checks. [Table 10-1](#) summarizes the checks performed for each stage.

Table 10-1 Checks Performed by the `check_block_abstraction` Command

Checks	Default	pre_place_opt	pre_clock_opt	pre_route_opt
CEL FRAM consistency	x	x	x	x
PG boundary pin consistency and PG pin connection	x	x	x	x
Don't touch attribute	x	x	x	x
Block leaf cell location	x	x	x	x
Multicorner-multimode	x	x	x	x
Block location		x	x	x
Clock tree synthesis			x	x

Table 10-1 Checks Performed by the check_block_abstraction Command (Continued)

Checks	Default	pre_place_opt	pre_clock_opt	pre_route_opt
Parasitic data				x
Signal integrity aggressor data				x

Applying Top-Level Constraints

Before you optimize a top-level design that contains block abstractions, perform the following tasks:

- Apply the full-chip SDC constraints by using the `read_sdc` command.
 - Apply the top-level UPF constraints by using the `load_upf` command.
- Block-level UPF constraints are stored as part of the block abstractions and are loaded during linking.
- Apply the full-chip clock tree exceptions.

Ensure that the exceptions for clock objects within the block abstractions are consistent with the block-level exceptions used when performing clock tree synthesis at the block level.

Note:

When you apply the full-chip SDC constraints or clock tree exceptions at the top level, you might get error and warning messages because some constraints are meant for objects within the hierarchical blocks that do not exist in the corresponding block abstractions. You must verify that each error and warning message can be safely ignored. To prevent these error and warning messages, modify these constraints to contain only those that are applicable to the objects retained in the block abstractions.

Controlling the Extent of Logic Loaded for Block Abstractions

When you create a block abstraction with the `create_block_abstraction` command, in addition to identifying and retaining the interface logic, the tool identifies the timing-critical paths for every input and output of the block abstraction. As a result, you can control the extent of logic that is loaded for a block abstraction by using the `-load_logic` option with the `set_top_implementation_options` command as follows:

- To load the entire block abstraction, consisting of all the interface logic, use the `-load_logic full_interface` option setting, which is the default.
- To load a compact version of the block abstraction, consisting of only the logic for the timing-critical paths of every input and output, use the `-load_logic compact_interface` option setting.

When you perform timing analysis at the top level, you can use the compact interface for block abstractions. For blocks with ports that fan out to a large number of boundary registers, using the compact interface can improve runtime and memory usage.

When you perform optimization, you can use the compact interface for the block abstractions that you do not want to optimize. However, you must use the full interface for the block abstractions that you want to optimize using transparent interface optimization.

Note:

If you switch a block abstraction from the compact interface to the full interface, you must reload the timing constraints.

When you create a block abstraction, the tool identifies the critical paths for the compact interface based on the current timing constraints and settings. Therefore, use the same timing constraints and settings when you use the compact interface for timing analysis and optimization at the top level.

Ignoring Timing Paths Entirely Within Block Abstractions

A block abstraction can contain register-to-register paths that are entirely within the block. Such paths are the result of combinational logic in the block abstraction that is shared by input-to-register and register-to-output paths.

To ignore register-to-register paths that are entirely within block abstractions, use the following variable setting:

```
icc_shell> set_app_var timing_ignore_paths_within_block_abstraction true
```

This variable setting does not affect paths that begin within a block abstraction, traverse outside, and return to the block abstraction.

Because paths entirely within blocks are optimized during block-level optimization, ignoring them during top-level optimization enables transparent interface optimization to focus on other paths.

Using Block Abstractions in a Multicorner-Multimode Flow

The IC Compiler multicorner-multimode flow supports hierarchical designs with blocks that are modeled using block abstractions.

By default, the multicorner-multimode flow supports the following hierarchical design characteristics:

- A top level and blocks that do not have scenarios
- A top level and blocks that have scenarios with identical names
- A top level that has scenarios and blocks that do not have scenarios
- Blocks that have additional scenarios that are not used during the top-level implementation
- TLUPlus models and temperature corners that are used during block-level implementation that match the TLUPlus models and temperature corners used at the top level

Note:

You can have additional TLUPlus models and temperatures at the top level. However, this could lead to inaccurate results. Therefore, use it only in the early stages of the design.

If the scenario names at the top level do not match the scenario names of the blocks, map the scenarios at the top level to the scenarios within the blocks by using the `select_block_scenario` command, as shown in the following example:

```
icc_shell> select_block_scenario -scenarios top_scen \
           -block_references blk -block_scenario blk_scen
```

If there are additional scenarios at the top level that do not exist in the blocks, for the most accuracy, define the additional scenarios for the blocks and re-create the block abstractions. However, in the initial stages of design development, you can use the existing block abstraction after you map the additional scenarios at the top level to the scenarios within the blocks by using the `select_block_scenario` command.

To reset the scenario mappings you have specified, use the `-reset` option with the `select_block_scenario` command.

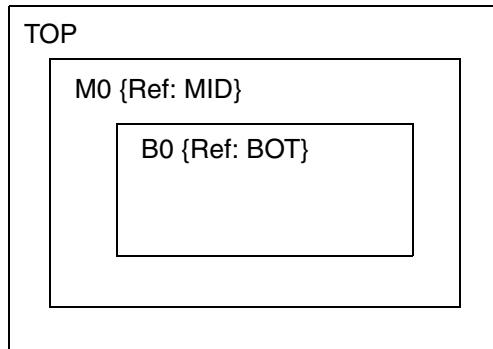
For more information about multicorner-multimode support, see “[Setting Up for Multicorner-Multimode Analysis and Optimization](#)” on page 3-33.

Handling Nested Physical Hierarchy

You can use the `create_block_abstraction` command to create block abstractions for blocks at any level of a design with nested physical hierarchy. When you create the block abstractions, start at the lowest level and move up the hierarchy.

You can run the `report_block_abstraction` and `check_block_abstraction` commands at any level of the nested physical hierarchy.

Figure 10-2 Block Abstraction With Nested Physical Hierarchy



For multicorner-multimode designs, if necessary, specify the scenario mapping from the top level to each of the levels by using the `select_block_scenario` command. For more information about specifying multicorner-multimode settings, see “[Using Block Abstractions in a Multicorner-Multimode Flow](#)” on page 10-10.

You can run transparent interface optimization at any level of a nested physical hierarchy. Before you run transparent interface optimization on a design with nested block abstractions, you must specify the references of the blocks instantiated in all levels of the hierarchy, as shown in the following example:

```
icc_shell> set_top_implementation_options -block_references {MID BOT}
```

For more information about running transparent interface optimization, see “[Performing Transparent Interface Optimization](#)” on page 10-13.

Viewing Block Abstractions in the GUI

In the IC Compiler GUI, a block abstraction is displayed as a level of physical hierarchy: a rectangular or rectilinear shape with a fill pattern. The View Settings panel provides additional controls over the block abstraction display.

- To display the block abstraction pins, select the Pin visibility option.
- To expand the block abstraction to show the leaf cells, pins, nets, and macros within it, type or select a positive integer in the “View level” box, click the Settings tab, and then select CEL in the Child View Name list.

For a block abstraction with more than one level of physical hierarchy, set the “View level” to the level of hierarchy up to which you want to expand the block abstraction.

You can select and highlight the block abstraction objects, such as leaf cells, pins, and nets of selected objects, in the layout view.

- To reset the display to the default, unexpanded view, type or select 0 in the “View level” box.

Performing Top-Level Optimization

To perform top-level placement, optimization, clock tree synthesis, and routing, the blocks represented by block abstractions must undergo placement, optimization, clock tree synthesis, and routing at the block level before the corresponding block abstractions are created.

You perform top-level placement and optimization by using the `place_opt` command. During top-level placement and optimization, the tool treats the cells within the block abstractions as fixed cells.

You perform top-level clock tree synthesis and optimization by using the `clock_opt` command. For more information about clock tree synthesis and optimization at the top level, see “[Handling Hierarchical Designs](#)” on page 5-6.

You perform top-level routing and postroute optimization by using the `route_opt` command. At the top level, the tool treats the block abstractions as placement blockages, while the available routing resources over the block abstractions are obtained from the corresponding FRAM views.

The detailed parasitic information stored with block abstractions is used during top-level routing, postroute optimization and extraction. For consistency, use the `set_extraction_options` command to apply the same postroute extraction scaling factors at the top level as those used at the block level.

Performing Transparent Interface Optimization

A top-level flow using block abstractions allows optimization of only the top-level logic. This is insufficient for interface timing when

- The top-level design has limited scope for optimization, such as in a design with an abutted or narrow-channel floorplan.
- Budgeting is incomplete or inaccurate, causing a jump in interface path delays when the blocks are assembled at the top level.
- Changes to blocks are needed in the postroute stage, making timing closure difficult.

Transparent interface optimization is a process that optimizes the interface logic within the blocks during top-level postroute optimization and thus achieves timing closure.

Topology-based postroute optimization performed by using the `route_opt` and `focal_opt` commands supports interface optimization of the blocks.

The following `route_opt` command options support interface optimization:

- `-effort`
- `-skip_initial_route`
- `-incremental`
- `-power`
- `-size_only`
- `-area_recovery`
- `-only_hold_time`
- `-only_design_rule`
- `-only_power_recovery`
- `-only_area_recovery`

The following `route_opt` command options do not support interface optimization:

- `-xtalk_reduction`
- `-only_xtalk_reduction`
- `-optimize_wire_via`
- `-wire_size`

- `-only_wire_size`
- `-stage global | track`

The following `focal_opt` command options support interface optimization:

- `-effort`
- `-setup_endpoints`
- `-hold_endpoints`
- `-drc_nets`
- `-drc_pins`
- `-size_only_mode`
- `-prioritize`

The following `focal_opt` command options do not support interface optimization:

- `-xtalk_reduction_nets`
- `-power`

During transparent interface optimization, the tool

- Optimizes both top-level logic and block interface logic during topology-based optimization
- Performs sizing optimization on UPF objects in multivoltage designs, such as isolation cells, level-shifter cells, and always-on cells that drive the always-on nets
- Updates the in-memory representation of the top-level and block netlists after topology-based optimization
- Runs legalization, ECO routing, and block abstraction on each of the updated blocks
- Relinks the top-level design with the updated blocks and runs the `report_qor` command at the top level
- Overwrites the Milkyway design (CEL view) of the blocks present in the top-level design library with all the changes that occurred during optimization

Setting Up for Transparent Interface Optimization

To set up for transparent interface optimization,

1. (Optional) If the top level and the block abstractions have different site row configurations, set the following variable before you open your design:

```
icc_shell> set_app_var tio_reload_block_site_rows true
```

2. Define the distributed processing configuration to use when updating the blocks after transparent interface optimization by using the `set_host_options` command.

For more information about setting up the distributed processing environment for the block update phase, see “[Setting Up the Distributed Processing Environment for Block Updates](#)” on page 10-19.

Note:

You can enable multicore processing for top-level optimization also. For more information about enabling multicore processing “[Enabling Multicore Processing](#)” on page 2-12

3. Copy the Milkyway design for each block you want to optimize to the top-level design library by using the `copy_mw_cel` command.

The blocks you want to optimize must be detail routed and be present in the same design library as the top-level design before you can perform interface optimization.

4. (Optional) Prevent optimization of specific objects within the blocks by using the `set_dont_touch` command from the top level to set `dont_touch` attributes on these objects.

During transparent interface optimization, the tool does not honor the `dont_touch` attributes you previously applied by using the `set_dont_touch` command at the block level.

5. (Optional) Enable transparent interface optimization for designs with multiply instantiated blocks by using the following variable setting:

```
icc_shell> set_app_var tio_allow_mim_optimization true
```

When you run transparent interface optimization, the tool inserts, deletes, and sizes cells inside a multiply instantiated block only if it does not degrade the QoR of all instances of the block.

6. Specify the following top-level implementation options by using the `set_top_implementation_options` command:
 - Enable interface optimization by using the `-optimize_block_interface true` option.
 - Specify the list of blocks whose interface logic you want to optimize by using the `-block_references` option.
 - Identify the distributed processing configuration to use for the block update phase by using the `-host_options` option.
7. Check the interface optimization setup by using the `check_interface_optimization_setup` command.
8. (Optional) If necessary, specify a different target library for a block by using the `set_target_library_subset` command.

For example, to set the target libraries for the design to lib1.db and lib2.db, but restrict the block U1 to cells from library lib2.db only, use the following commands:

```
icc_shell> set_target_library "lib1.db lib2.db"
icc_shell> set_target_library_subset "lib2.db" \
    -object_list [get_cells u1]
```

The following example shows the sequence of commands used to set up for interface optimization:

```
icc_shell> set_host_options -name for_tio -num_processes 4
icc_shell> copy_mw_cel -from_library $block_ref_lib -from $blocks
icc_shell> set_top_implementation_options -host_options for_tio \
    -optimize_block_interface true -block_references $blocks
icc_shell> check_interface_optimization_setup
```

Specifying Block-Specific Size-Only Settings

Use the `-size_only_mode` option with the `set_top_implementation_options` command to restrict the scope of optimization inside the blocks to only sizing-based changes. The sizing can further be restricted to in-place and footprint sizing.

Different settings can be specified for different blocks. Block-specific size-only settings are ignored when you use the `route_opt -size_only` command.

Customizing the Commands Executed During Block Update

You can specify a setup file to be sourced during the block update phase by using the `-block_update_setup_script` option of the `set_top_implementation_options` command. You can also specify a script file containing the set of commands to be executed during the block update phase by using the `-block_update_cmd_script` option. You can specify the same file for all the blocks or you can specify a unique file for each block.

By default, the tool executes the `legalize_placement` and `route_zrt_eco` commands during the block update phase. If you specify the `-block_update_cmd_script` option, the tool sources the script you specify instead of executing the `legalize_placement` and `route_zrt_eco` commands.

Note:

The tool always runs the `create_block_abstraction` command during the block update phase, even if you specify the `-block_update_cmd_script` option. However, if you include the `create_block_abstraction` command within the script specified with the `-block_update_cmd_script` option, the tool does not execute the `create_block_abstraction` command twice.

You can specify the setup or command file name using the absolute path name, as shown in the following example:

```
icc_shell> set_top_implementation_options \
-block_update_cmd_script /u/my_dir/my_blk1_update_scr.tcl
-block_references blk1
```

Alternatively, you can specify only the file name if you include the directory where it is located in the search path, as shown in the following example:

```
icc_shell> set_app_var search_path "$search_path /u/my_dir"
icc_shell> set_top_implementation_options \
-block_update_cmd_script my_blk1_update_scr.tcl
-block_references blk1
```

Preserving Relative Placement Groups Inside Hierarchical Blocks

The IC Compiler tool automatically prevents the cells and nets of the relative placement groups inside hierarchical blocks from being optimized during transparent interface optimization. However, the tool can move the cells in the relative placement group when performing legalization during the block update phase. To prevent this, use the following steps:

1. Create a customized block update command script and include the following command:

```
set_attribute [rp_group_references] is_fixed true
```

2. Specify the customized block update command script by using the `set_top_implementation_options -block_update_cmd_script` command.

Preserving the Route Properties of Nets Inside Hierarchical Blocks

To preserve the route properties of the nets inside the hierarchical blocks that are optimized during transparent interface optimization, set the `tio_preserve_routes_for_block` variable to `true`. The route properties of the optimized nets help guide the ECO routing stage of transparent interface optimization, which can improve timing convergence.

Skipping Automatic Block Updates

When you perform transparent interface optimization by using the `route_opt -incremental` command, you can skip the block update phase by using the following variable setting:

```
icc_shell> set_app_var tio_skip_block_update true
```

Skipping the automatic block update invalidates the parasitic data for the block and the results of any subsequent timing analysis at the top level. Therefore, update the blocks before you perform timing analysis.

If you perform timing analysis before updating the blocks, the tool issues a TL-410 error message. If you skip the block update phase, the tool issues a TL-410 error message when it generates a QoR report at the final stage of the `route_opt` command. To avoid the TL-410 error message when running the `route_opt` command, you can skip the QoR report by using the following variable setting:

```
icc_shell> set_app_var routeopt_skip_report_qor true
```

Generating ECO Files

To generate an ECO file containing all the netlist changes that occur during transparent interface optimization, for each block that is enabled for transparent interface optimization, use the following variable setting:

```
icc_shell> set_app_var tio_write_eco_changes true
```

By default, the tool writes the ECO files into a directory named TIO_eco_changes. If a directory by that name exists the tool adds a suffix of .1 to the directory name, and it increments the suffix by one each time it performs a subsequent optimization run.

To change the name of the directory into which the ECO file are written, use the `tio_eco_output_directory` variable. If you specify an output directory using the `tio_eco_output_directory` variable, the directory name is not incremented using a suffix for each subsequent optimization run. The tool uses the same directory, and appends to the existing ECO files.

Setting Up the Distributed Processing Environment for Block Updates

At the end of transparent interface optimization the tool updates each block opened for interface optimization, by performing legalization, ECO routing, and block abstraction.

To enable multicore execution of legalization, ECO routing, and block abstraction, use the following command:

```
set_host_options -name for_tio -max_cores N
```

To execute legalization, ECO routing, and block abstraction in parallel on various blocks in a distributed processing environment, use one of the following three methods:

- Specify a queuing system such as GRD or LSF:

```
set_host_options -name for_tio -pool lsf | grd  
-submit_options options
```

Ensure that your UNIX search path has a path to either the `qsub` or `bsub` command. Define all the settings in the `.cshrc` file that is associated with the queuing system.

Ensure that the settings related to the queuing system are defined before statements similar to the following:

```
if ($?USER == 0 || $?prompt == 0) exit
```

If you use UNIX shells other than `csh`, you must update the appropriate initialization file.

- Specify a list of named hosts:

```
set_host_options -name for_tio {host1 host2 ... hostH}
```

By default, the tool uses `rsh` as the submission program. To use `ssh` as the submission program, use the following command:

```
set_host_options -name for_tio host_name -submit_command ssh
```

Set up the environment such that `ssh` works without requiring a password.

- Use multiple processes on the same host:

To launch separate processes on the same host without using `rsh` or `ssh`, use one of the following 3 options:

- `set_host_options -name for_tio -num_processes N`
- `set_host_options -name for_tio -num_processes N localhost`
- `set_host_options -name for_tio -num_processes N [sh host_name]`

Reporting the Transparent Interface Optimization Settings

Use the `report_top_implementation_options` command to verify the options that are set by using the `set_top_implementation_options` command. The command reports the options in the following format:

Block Reference	Optimize Interface	Optimize Shared Logic	Size Only Mode	Host Options
block1	true	false	off	my_hosts
block2	true	false	off	my_hosts

Example Scripts

The following example scripts perform various tasks related to block abstractions:

Example 10-1 Script to Create Block Abstractions

```
set blocks "block1 block2"
foreach block $blocks {
    open_mw_cel $block
    create_block_abstraction
    save_mw_cel
    close_mw_cel
}
```

Example 10-2 Script to Run the Top-Level Flow With Block Abstractions

```

set blocks "block1 block2"
open_mw_cel TOP
# you must run the following command before using the GUI or collections
# or linking the design
set_top_implementation_options -block_references $blocks \
    -load_logic compact_interface
foreach block $blocks {
    change_macro_view -reference $block -view FRAM
}
read_sdc full_chip.sdc
place_opt
clock_opt
route_opt

```

Example 10-3 Script to Run the Top-Level Flow With Block Abstractions and Transparent Interface Optimization

```

set blocks "block1 block2 block3"
set blocks_to_optimize "block1 block2"
open_mw_cel TOP
# You must run the following command before using the GUI or collections
# or linking the design
set_top_implementation_options -block_references $blocks \
    -load_logic compact_interface
foreach block $blocks {
    change_macro_view -reference $block -view FRAM
}
read_sdc full_chip.sdc
place_opt
clock_opt
close_mw_cel -save
#
# You must copy the Milkyway designs for the blocks you want to optimize
# to the top-level design library
copy_mw_cel -from_library block1_ref_lib -from block1
copy_mw_cel -from_library block2_ref_lib -from block2
open_mw_cel TOP
set_host_options -max_cores 4
set_host_options -name for_tio -max_cores 4 -pool grd \
    -submit_options "-l mem_avail=1G,mem_free=1G"
set_top_implementation_options -host_options for_tio \
    -optimize_block_interface true -block_references $blocks_to_optimize \
    -load_logic full_interface
# You must reload the SDC constraints because you are changing the
# block abstractions from the compact interface to the full interface.
read_sdc full_chip.sdc
check_interface_optimization_setup
route_opt
save_mw_cel

```


11

In-Design Rail Analysis

The PrimeRail rail analysis capability is integrated with the IC Compiler implementation flow through the In-Design rail analysis interface. You can use In-Design 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 In-Design rail analysis to verify the robustness of the power and ground network. When placement is complete, use In-Design rail analysis to perform voltage drop analysis to calculate static power consumption and to check for voltage drop violations. You can also perform voltage drop analysis at other stages in the design flow, such as after detail routing. When chip finishing is complete, use In-Design rail analysis to perform electromigration analysis to check for current density violations.

This chapter includes the following sections:

- [Overview](#)
- [Prerequisites for In-Design Rail Analysis](#)
- [Preparing for In-Design Rail Analysis](#)
- [Performing Rail Integrity Checking](#)
- [Performing Voltage Drop Analysis](#)
- [Performing Electromigration Analysis](#)

- Performing Inrush Current Analysis
- Performing Minimum Path Resistance Analysis

Overview

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

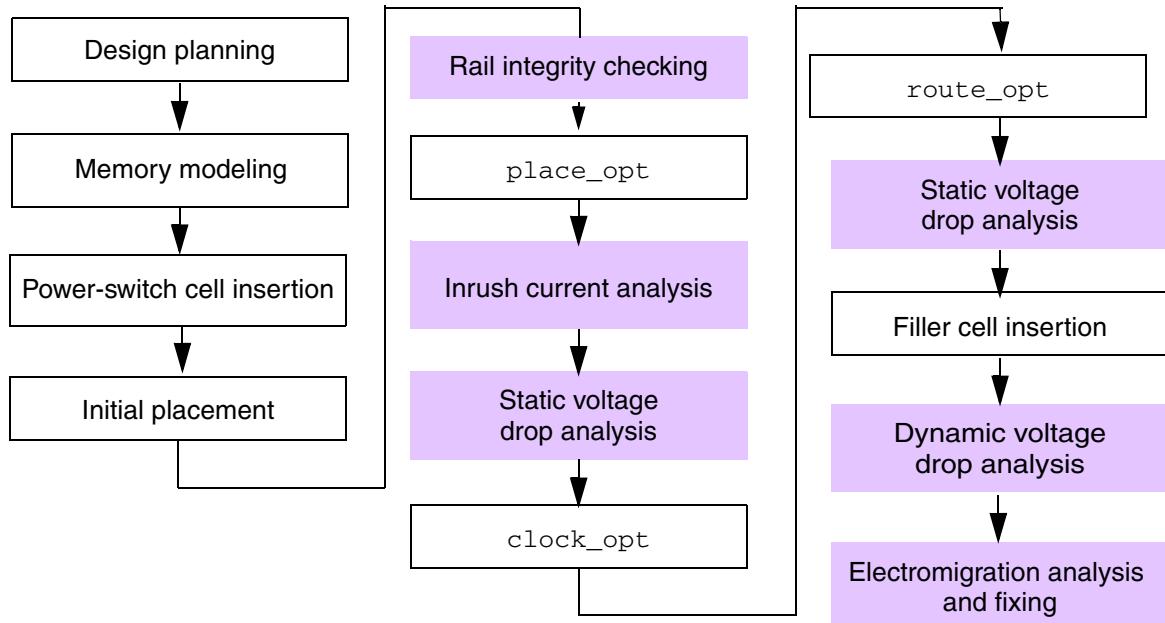
- Rail integrity checking
- Static rail analysis
 - Static voltage drop analysis
 - Static electromigration analysis
- Dynamic rail analysis
 - Dynamic voltage drop analysis
 - Dynamic electromigration analysis
 - Vector-free rail analysis
- Inrush current analysis
- Minimum path resistance analysis

You can perform In-Design rail analysis on both multicorner-multimode and non-multicorner-multimode designs; however, when analyzing a multicorner-multimode design, In-Design rail analysis 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 GUI, as well as error views that you can display in the IC Compiler error browser. These maps and error views help you to discover problem areas and determine corrective action without leaving the IC Compiler environment.

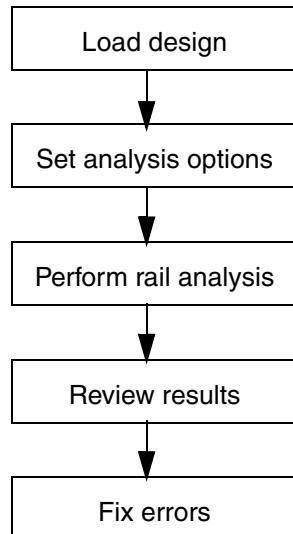
[Figure 11-1](#) shows where you would use these analysis capabilities in the typical design flow.

Figure 11-1 Using In-Design Rail Analysis in the IC Compiler Design Flow



[Figure 11-2](#) shows the basic flow for running In-Design rail analysis.

Figure 11-2 Basic In-Design Rail Analysis Flow



You use the `set_rail_options` command (or choose Rail > Set Rail Options in the GUI) to set the rail analysis options. You can set environment options, input options, and analysis options. For information about setting the rail analysis options, see “[Setting the Rail Analysis Options](#)” on page 11-9.

You can use In-Design rail analysis to perform the following types of rail analysis:

- Rail integrity checking

To perform rail integrity checking, you must first specify the rail integrity strategy by using the `set_rail_integrity_layout_check_strategy`, and then perform the check by using the `verify_rail_integrity` command. To perform these tasks in the GUI, choose Rail > Verify Rail Integrity.

To review the detected errors, display the generated error views in the error browser.

For more information about performing rail integrity checking, see “[Performing Rail Integrity Checking](#)” on page 11-18.

- 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.

To review the results, view the generated maps by using the `read_rail_maps` command (or choose the appropriate map from the Rail menu in the GUI) or display the generated error views in the error browser.

For more information about performing voltage drop analysis, see “[Performing Voltage Drop Analysis](#)” on page 11-32.

- Electromigration analysis

To perform electromigration analysis, use the `analyze_rail -electromigration` command or choose Rail > Analyze Rail and select “Electromigration analysis” in the GUI.

To review the results, view the generated maps by using the `read_rail_maps` command (or choose the appropriate map from the Rail menu in the GUI) or display the generated error views in the error browser.

For more information about performing voltage drop analysis, see “[Performing Electromigration Analysis](#)” on page 11-39.

- Inrush current analysis

To perform inrush current analysis, use the `analyze_rail -inrush` command or choose Rail > Analyze Rail and select “In-rush current analysis” in the GUI.

To review the results, view the generated maps by using the `read_rail_maps` command (or choose the appropriate map from the Rail menu in the GUI).

For more information about performing inrush current analysis, see “[Performing Inrush Current Analysis](#)” on page 11-43.

- Minimum path resistance analysis

To perform minimum path resistance analysis, use the `analyze_rail -min_path_resistance` command or choose Rail > Analyze Rail and select “Min Path Resistance analysis” in the GUI.

To review the results, view the generated map by using the `read_rail_maps` command (or choose the appropriate map from the Rail menu in the GUI).

For more information about performing minimum path resistance analysis, see “[Performing Minimum Path Resistance Analysis](#)” on page 11-47.

Prerequisites for In-Design Rail Analysis

Before you can perform In-Design rail analysis, ensure that you have the correct licenses and libraries, as described in the following sections.

License Requirements

At a minimum, you must have the following license keys to invoke the PrimeRail tool within the IC Compiler tool:

Feature	License keys required
Rail integrity checking	PrimeRail (or PrimeRail-static), In-Design-Rail-Integrity
Static analysis	PrimeRail (or PrimeRail-static), In-Design-Static-Rail
Dynamic analysis	PrimeRail (or PrimeRail-adv) , In-Design-Dynamic-Rail

Library Requirements

Before you run In-Design rail analysis, you must ensure that the libraries meet the following requirements:

- The logic libraries contain power data

Rail analysis requires logic libraries characterized with power data. You can use either CCS power or nonlinear power model (NLPM) libraries. CCS power libraries are required for dynamic rail analysis and are recommended for all other rail analysis types.

For more information about power modeling, see the *Library Compiler Timing, Signal Integrity, and Power Modeling User Guide*.

- The power-related attributes are defined in the technology file
 - Verify that the following power-related units are defined in the Technology section:
 - The voltage unit and precision, using the `unitVoltageName` and `voltagePrecision` attributes
 - The current unit and precision, using the `unitCurrentName` and `currentPrecision` attributes
 - The power unit and precision, using the `unitPowerName` and `powerPrecision` attributes
 - Verify that the upper and lower layers are defined in the ContactCode section for the polysilicon-contact, contact, and via layers. The polysilicon-contact layer is the layer between polysilicon and metal. The contact layer is the layer between diffusion and metal. The via layer is between metal N and metal N+1.

For more information about the technology file, see the *IC Compiler Technology File and Routing Rules Reference Manual*.

- The TLUPlus files are specified

In-Design rail analysis uses TLUPlus models for analyzing resistance and capacitance effects on power and ground nets when performing voltage drop analysis or electromigration analysis. To set up the TLUPlus files, use the `set_tlu_plus_files` command.

For more information about specifying the TLUPlus files, see the “[Setting Up the TLUPlus Files](#)” on page 3-40.

- The library robustness for In-Design rail analysis has been verified

To check the robustness of the Milkyway design library for rail analysis, use the `check_library` command, as shown in the following example:

```
icc_shell> set_check_library_options -rail view_data  
icc_shell> check_library -mw_library_name my_mw_lib
```

When you enable the rail analysis checks, the `check_library` command checks for rail data consistency between the CONN and FRAM views in the Milkyway design library. If the `check_library` command finds mismatches for the PG port type, net type, or number, it generates warning messages.

For more information about the `check_library` command, see the *Library Data Preparation for IC Compiler User Guide*.

- (Optional) The on-state resistance values are defined for the power-switch cells

If the design contains power-switch cells to reduce leakage current and you want to run static rail analysis in the power-on mode, you must ensure that the on-state resistances of the power-switch cells are defined.

If the logic libraries have I-V curves for the power-switch cells, In-Design rail analysis automatically determines the on-state resistance values from the I-V curve table. [Example 11-1](#) shows an I-V curve table that specifies the on-state resistances of the power-switch cells.

Example 11-1 I-V Curve Table With On-State Resistance Data

```
dc_current ( ivt1 ) {
    related_switch_pin : SLEEP; /* control pin */
    related_pg_pin : VDD; /* source */
    related_internal_pg_pin : VVDD; /* drain */
    values( "0.010, 0.020, 0.030, 0.030, 0.030", \
            "0.011, 0.021, 0.031, 0.041, 0.051", \
            "0.012, 0.022, 0.032, 0.042, 0.052", \
            "0.013, 0.023, 0.033, 0.043, 0.053", \
            "0.014, 0.024, 0.034, 0.044, 0.054");
}
```

If the logic libraries do not have I-V curves for the power-switch cells, you can explicitly specify the on-state resistance values by using the PrimeRail `set_mtcmos_switch_lib_cell` command in the PrimeRail setup file, `.synopsys_pr.setup`. This command sets switching-related library values for power-switch reference cells, including the on-state resistance, off-state leakage, and pin state name. In-Design rail analysis uses this information to calculate voltage drop and current violations.

For example, to set the on-state resistance value for the HEAD2X16_HVT power-switch cell, add the following command to the `.synopsys_pr.setup` file.

```
set_mtcmos_switch_lib_cell HEAD2X16_HVT -input_pg_pin VDDG \
    -output_pg_pin VDD -pin_state { SLEEP 0 } \
    -pin_state_name FULL_ON -on_state_resistance 1
```

For more information about the library requirements for rail analysis, see the related chapter in the *PrimeRail User Guide*.

Preparing for In-Design Rail Analysis

To prepare for In-Design rail analysis, you must perform the following tasks:

- Set the rail analysis options
- For multicorner-multimode designs, set the scenario for analysis
- (Optional) Enable multicore processing

The following sections describe these tasks.

Setting the Rail Analysis Options

Before running In-Design rail analysis, you must use the `set_rail_options` command to set the rail analysis options.

To report the current settings of the rail analysis options, use the `report_rail_options` command.

To reset all the rail analysis options to their default settings, use the `set_rail_options -default` command.

The rail analysis options are not saved in the design database. To save the rail analysis options for use in a subsequent session, use the `write_rail_options` command. When you run this command, you must use the `-output` option to specify the name of the generated script file. To import the saved settings, use the `source` command to load the script file.

The following sections describe some commonly used options:

- [Specifying the PrimeRail Executable File](#)
- [Specifying the Working Directory](#)
- [Setting the Analysis Mode](#)
- [Defining the Ideal Voltage Sources](#)
- [Setting the Operating Voltage](#)
- [Controlling Pin Shape Derivation During PG Extraction](#)

For a summary of the available options, see “[Summary of the Rail Analysis Options](#)” on page 11-13.

Specifying the PrimeRail Executable File

By default, In-Design rail analysis searches for the PrimeRail executable file by using your Linux or UNIX PATH environment variable.

To explicitly specify the location of the PrimeRail executable file, use the `-pr_exec_dir` option with the `set_rail_options` command to specify the path.

Specifying the Working Directory

During rail analysis, In-Design rail analysis creates a working directory to store the generated files. By default, In-Design rail analysis creates a working directory named `pr_design`, where `design` is the top-level design in the current Milkyway cell.

To use a different name for the working directory, use the `-output_dir` option with the `set_rail_options` command to specify the name.

Setting the Analysis Mode

By default, In-Design rail analysis performs static rail analysis. To specify a different mode, use the `-analysis_mode` option with the `set_rail_options` command.

- To perform static voltage-drop or electromigration analysis, set the analysis mode to `static`.
- To perform dynamic voltage-drop or electromigration analysis, set the analysis mode to `dynamic`.
- To perform dynamic analysis without activity vectors, set the analysis mode to `dynamic_vectorfree`.

By default, the tool determines the number of clock cycles and the simulation time used for the vector-free analysis. To specify the number of clock cycles, use the `-dynamic_vectorfree_cycles` option. To specify the simulation time in nanoseconds, use the `-dynamic_vectorfree_sim_time` option.

- To perform inrush current analysis, set the analysis mode to `inrush_only`.

When you use this mode, the tool performs only inrush analysis. You can also run inrush current analysis in the dynamic analysis mode. In this case, the tool performs combined dynamic and inrush analysis.

Defining the Ideal Voltage Sources

To define the ideal voltage sources (taps), use one of the following options with the `set_rail_options` command:

- `-pad_master_file`

The pad master file specifies the pad master cells to be used as ideal voltage sources for rail analysis. You can use this option to support bump cells when pins are unavailable or are defined as signal pins.

Use the following syntax to specify the pad master cells:

```
master_name [net_name] [R(resistance)] [L(inductance)] [C(capacitance)]
```

If you specify a supply net name, the cell is treated as an ideal voltage source only when analyzing that supply net. Otherwise, the specified pad master is used for all the supply nets being analyzed.

You can optionally specify the resistance (R), inductance (L), and capacitance (C) values to be used for these ideal voltage sources during rail analysis.

For example, to use the VDD.FRAM pad master as an ideal voltage source for the VDD net, use the following statement:

```
VDD.FRAM VDD
```

If the specified pad cells have non-PG pins that are connected to a supply net, by default, the tool does not use these pins as ideal voltage sources. To allow these pins to be used as ideal voltage sources, set the

`-allow_non_pg_pin_for_voltage_source_creation` option to true.

- `-pad_instance_file`

The pad instance file specifies the pad cell instances to be used as ideal voltage sources for rail analysis. You can use this option to support bump cells when pins are unavailable or are defined as signal pins.

Use the following syntax to specify the pad cell instances:

```
instance_name [net_name] [R(resistance)] [L(inductance)] [C(capacitance)]
```

If you specify a supply net name, the cell is treated as an ideal voltage source only when analyzing that supply net. Otherwise, the specified pad instance is used for all the supply nets being analyzed.

You can optionally specify the resistance (R), inductance (L), and capacitance (C) values to be used for the ideal voltage source during rail analysis.

For example, to use the VDD1 pad instance as an ideal voltage source for the VDD net, use the following statement:

```
VDD1 VDD
```

If the specified pad cells have non-PG pins that are connected to a supply net, by default, the tool does not use these pins as ideal voltage sources. To allow these pins to be used as ideal voltage sources, set the

`-allow_non_pg_pin_for_voltage_source_creation` option to `true`.

- `-user_defined_tap_file`

The tap file specifies the physical location (coordinates and layer numbers) of the ideal voltage sources to be used for rail analysis. You can use this option when you are working with a block or a design that does not yet have pins defined, or the location or the design of the pad cells is not yet finalized.

Use the following syntax to specify the physical location of the ideal voltage sources, where the x- and y-coordinates are in microns:

```
net_name layer_number x y [R(resistance)] [L(inductance)] [C(capacitance)]
```

You can optionally specify the resistance (R), inductance (L), and capacitance (C) values to be used for the tap during rail analysis.

If the location specified for a tap point does not touch a layout shape, the tool issues a warning message. To enable the tool to find a substitute location, use the `-snap_distance` option to specify the distance in microns within which to search for a layout shape. When you specify this option, the tool looks for a layout shape on the specified layer that is within the snap distance of the specified tap location, either horizontally or vertically. The tool then reinserts the tap at the corner of a rectangle that is within the minimum distance to the specified location.

- `-use_pins_as_pads`

To use top-level power and ground pins as ideal voltage sources during rail analysis, set this option to `true`. By default, this option is `false` and the tool does not use top-level power and ground pins as ideal voltage sources.

- `-virtual_pads`

To use the virtual pads defined during power network analysis and power network synthesis as ideal voltage sources during rail analysis, set this option to `true`. By default, this option is `false` and the tool does not use virtual pads as ideal voltage sources.

Setting the Operating Voltage

By default, In-Design rail analysis uses the operating voltage specified with the `set_voltage` command for rail, power, and timing analysis.

To specify a different operating voltage to use for rail analysis, use the `-supply_voltage` option with the `set_rail_options` command.

The syntax to specify the operating voltage is

```
-supply_voltage { "supply_name root_voltage [operating_voltage]" ... }
```

If you specify two voltage values, the first value is used for rail analysis and the second is used for power and timing analysis. If you specify only one voltage value, it is used only for rail analysis; power and timing analysis use the voltage set by the `set_voltage` command.

If a supply net does not have an operating voltage assigned by either of these commands, the tool uses different methods to determine the voltage, depending on the type of analysis.

- For power and timing analysis, the tool determines the voltage from the operating condition settings.
- For rail analysis, the tool determines the voltage from the minimum voltage from all the cell instances connected to the supply net, considering the worst case.

For example, assume there are three cell instances that connect to the VDD supply net. One cell instance has a voltage of 1.0 and the other two cell instances have a voltage of 1.1. In this case, In-Design rail analysis uses 1.0 for VDD in rail analysis.

To check the voltage settings on the supply nets before running In-Design rail analysis, run the `report_supply_net` command.

Controlling Pin Shape Derivation During PG Extraction

By default, the `analyze_rail` command derives the pin shapes from the FRAM view during PG extraction. To use the pin shapes derived from the macro cells connected to the analyzed supply nets instead, use the `-ignore_fram_view_for_pg_extraction true` option with the `set_rail_options` command.

Summary of the Rail Analysis Options

Table 11-1 lists the commonly used options of the `set_rail_options` command.

Table 11-1 Commonly Used Options of the set_rail_options Command

Option	Description
<code>-pr_exec_dir</code>	Specifies the location of the PrimeRail executable file. By default, the IC Compiler tool locates the PrimeRail executable file by using the UNIX search path. For more information, see “ Specifying the PrimeRail Executable File ” on page 11-10.
<code>-output_dir</code>	Specify the output directory for the analysis results. By default, the output directory is named <code>./pr_design</code> . For more information, see “ Specifying the Working Directory ” on page 11-10.

Table 11-1 Commonly Used Options of the set_rail_options Command (Continued)

Option	Description
-supply_voltage	Specifies the ideal operating voltage for the supply net used for performing rail analysis in PrimeRail.
	For more information, see “ Setting the Operating Voltage ” on page 11-12.
-analysis_mode	Specifies the analysis mode to use.
	By default, the tool performs static rail analysis.
	For more information, see “ Setting the Analysis Mode ” on page 11-10.
-dynamic_vectorfree_cycles	Specifies the number of dominant clock cycles for which to simulate the design when performing dynamic vector-free analysis.
	If you do not specify this option, the tool determines the number of clock cycles.
-dynamic_vectorfree_sim_time	Specifies the time in nanoseconds for which to simulate the design when performing dynamic vector-free analysis.
	If you do not specify this option, the tool determines the simulation time.
-use_pins_as_pads	Controls whether the top-level power and ground pins are treated as ideal voltage sources for rail analysis and integrity analysis.
	For more information, see “ Defining the Ideal Voltage Sources ” on page 11-11.
-virtual_pads	Controls whether the virtual pads in the current IC Compiler session are used as the ideal voltage sources for rail analysis and integrity analysis.
	For more information, see “ Defining the Ideal Voltage Sources ” on page 11-11.
-pad_master_file	Specifies an optional pad master file, which specifies the pad master cells to be used as ideal voltage sources for rail analysis.
	For more information, see “ Defining the Ideal Voltage Sources ” on page 11-11.

Table 11-1 Commonly Used Options of the set_rail_options Command (Continued)

Option	Description
<code>-pad_instance_file</code>	Specifies an optional pad instance file, which specifies the pad cell instances to be used as ideal voltage sources for rail analysis. For more information, see “Defining the Ideal Voltage Sources” on page 11-11 .
<code>-allow_non_pg_pin_for_voltage_source_creation</code>	Allows the creation of taps on non-PG pins on pad cell pins connected to supply nets defined within either a pad master file or a pad instance file. For more information, see “Defining the Ideal Voltage Sources” on page 11-11 .
<code>-user_defined_tap_file</code>	Specifies an optional tap file, which defines the physical location (coordinates and layer numbers) of the ideal voltage sources to be used for rail analysis. For more information, see “Defining the Ideal Voltage Sources” on page 11-11 .
<code>-snap_distance</code>	Defines the snap distance in microns. The snap distance is used to search from the original tap location to the left, right, bottom, and top when a tap location does not touch a layout shape. For more information, see “Defining the Ideal Voltage Sources” on page 11-11 .
<code>-ignore_fram_view_for_pg_extraction</code>	Controls whether the FRAM view or connected macro cells are used to derive the pin shapes during PG extraction. For more information, see “Controlling Pin Shape Derivation During PG Extraction” on page 11-13 .
<code>-sdc</code>	Specifies an SDC script file. If you do not specify this option, the tool creates an SDC file based on the data saved in the Milkyway design.
<code>-spef</code>	Specifies a parasitic SPEF file. If you do not specify this option, the tool creates a parasitic SBPF file based on the data saved in the Milkyway design.

Table 11-1 Commonly Used Options of the set_rail_options Command (Continued)

Option	Description
-switching_activity	Specifies a switching activity file to be used in rail analysis. The switching activity file can be in VCD, SAIF, or Tcl format. If you do not specify this option, the default toggle rate is used. For more information, see “ Defining the Switching Activity Information ” on page 11-33.
-report_effective_voltage_drop	Controls whether the tool calculates and reports the effective voltage drop values. By default, the tool does not report the effective voltage drop values. For more information, see “ Enabling Effective Voltage Drop Reporting ” on page 11-34.
-timing_windows_only	When effective voltage drop reporting is enabled, controls whether the tool calculates the voltage drop values for all time steps or only when the cell is in transition. By default, the tool calculates the voltage drop values for all time steps. For more information, see “ Enabling Effective Voltage Drop Reporting ” on page 11-34.
-em_rule	Specifies an optional electromigration rules file. The electromigration rules file can be in ALF, TLUPlus, or technology file format. If you do not specify this option, the tool uses the electromigration rules stored with the Milkyway design library. If there are no rules stored in the Milkyway design library, the tool uses the electromigration rules stored in the technology file. For more information, see “ Defining the Electromigration Rules ” on page 11-39.
-reuse	Selects which steps of the flow to reuse from a previous analysis run in the same session. For more information, see “ Reusing Rail Analysis Data ” on page 11-35.

Table 11-1 Commonly Used Options of the set_rail_options Command (Continued)

Option	Description
-assign_power_only	<p>Specifies that In-Design rail analysis runs power analysis using the user-specified power values from a cell instance power file.</p> <p>By default, the option is set to <code>false</code> and In-Design rail analysis uses the power values generated by the integrated power analysis.</p> <p>Load the cell instance power file with the <code>set_rail_options -config_file</code> option.</p>
-config_file	<p>Specifies the name of the configuration file that defines additional configuration settings, such as user-defined power or voltage values.</p> <p>The file that describes the user-defined power values is called the cell instance power file. This file supports the following commands:</p> <pre>defineCellInstancePower cellId cellInstName value defineCellInstancePowerByPort cellId cellInstName \ portName value defineCellInstancePowerByMaster cellId \ cellMasterName value</pre>
-write_rail_data	<p>Writes rail data from current session into text files that can be used to set the rail data in another session. You can save the cell instance power data (<code>instance_power</code>), PG pin power data (<code>pg_pin_power</code>), or both.</p> <p>By default, the rail data is not saved.</p>
-top_level_only	<p>Controls whether the script file and necessary SBPF and SDC files are generated only for the top-level design. When enabled, no SBPF files are created for the subblocks. For more information, see “Defining the Analysis Scope” on page 11-34.</p>
-default	Resets all the options to their default settings.

Selecting the Scenario

When analyzing a multicorner-multimode design, In-Design rail analysis analyzes only the current scenario; therefore you must select or create a scenario that best represents the desired operating condition for rail analysis and set it as the current scenario.

Enabling Multicore Processing

In-Design rail analysis supports multithreading, which improves turnaround time by performing tasks in parallel more quickly than if they were run in serial on a single core.

By default, In-Design rail analysis uses four cores. To change the number of cores used for multithreading, use the `set_host_options` command with the `-max_cores` option. For more information about using the `set_host_options` command to configure multithreading, see “[Configuring Multithreading](#)” on page 2-12.

Performing Rail Integrity Checking

Rail integrity checking checks the physical connectivity of the power network structure. You can perform rail integrity checking after completing the power structure and before running the `place_opt` command.

Rail integrity checking can identify the following power network issues:

- Floating shapes

Floating shapes are terminals, vias, and net shapes that are not contiguously and physically connected to an ideal voltage source.

- Floating pin shapes

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

Note:

This check considers only those supply pins that are connected in the netlist; it does not verify the netlist connectivity of all supply pins.

- Dangling pin access edges

Dangling pin access edges are access edges of PG pins that do not touch any other same-net shapes but are connected to the ideal voltage source.

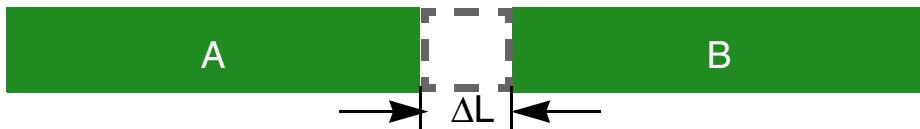
- Dangling vias

Dangling vias are vias that are not floating, but have one physically unconnected open-ended metal enclosure.

In most cases, dangling vias are the result of removing the upper or lower metal layer during routing. Dangling vias are unwanted vias and should be removed from the design.

- Missing vias
If two overlapping metal shapes do not contain a via within their overlapping area, it is considered a potential missing via error.
- Discontinuous connections
If two metal shapes (wires, paths, pins, or rectangles) are placed within a minimum distance of each other on the same supply net on the same layer, the gap between the shapes is identified as a discontinuous connection error.
For example, if the gap (ΔL) shown in [Figure 11-3](#) is smaller than the threshold, the gap is flagged as a discontinuous connection error.

Figure 11-3 Discontinuous Connection



For detailed information about these checks, see the *PrimeRail User Guide*.

To perform rail integrity checking,

1. Set the rail analysis options.

For details, see “[Setting the Rail Analysis Options](#)” on page 11-9.

2. Define the rail integrity layout check strategy, which defines the specifics of the checks to perform.

For details, see “[Defining the Rail Integrity Layout Check Strategies](#)” on page 11-20.

3. Save the design.

For details, see “[Saving the Design in Milkyway Format](#)” on page 3-60.

4. Run the rail integrity checking.

For details, see “[Running the Rail Integrity Checking](#)” on page 11-28.

The following sections describe how to perform these tasks.

Defining the Rail Integrity Layout Check Strategies

To enable checks during rail integrity checking, you must define one or more rail integrity layout check strategies.

To define the strategies, use the `set_rail_integrity_layout_check_strategy` command.

Note:

The strategies defined with the `set_rail_integrity_layout_check_strategy` command are available only in the current session; they are not saved in the design database. To save a strategy for use in a subsequent session, use the `write_rail_integrity_strategy` command.

To report a strategy from the current session, run the `report_rail_integrity_layout_check_strategy` command. To remove a strategy from the current session, use the `remove_rail_integrity_layout_check_strategy` command.

Alternatively, you can use the GUI to define the rail integrity layout check strategies and perform rail integrity checking. For information, see “[Performing Rail Integrity Checking in the GUI](#)” on page 11-30.

The following sections describe how to use the `set_rail_integrity_layout_check_strategy` command to define strategies for each of the rail integrity layout checks.

Defining a Strategy for the Floating Shapes Check

To enable the floating shapes check, use the `-non_ideal_sourced_floating_shapes` option with the `set_rail_integrity_layout_check_strategy` command.

If two shapes touch, intersect, or overlap each other on the same layer or between layers with via cut layers, the tool considers them as connected and assigns them the same group ID. If a group is not connected to a tap point, the tool reports all shapes inside this group as a set of related floating errors.

For information about how to control the tracing behavior used for this check, see “[Controlling the Tracing Behavior](#)” on page 11-27.

Defining a Strategy for the Floating Pin Shapes Check

In-Design rail analysis supports two types of floating pin checks:

- At least one electrically equivalent pin shape is not floating (minimal check)

To enable the minimal floating pin shape check, use the `-pg_pin_floating_connection_check minimal` option with the `set_rail_integrity_layout_check_strategy` command.

- All electrically equivalent pin shapes are not floating (maximal check)

To enable the maximal floating pin shape check, use the `-pg_pin_floating_connection_check maximal` option with the `set_rail_integrity_layout_check_strategy` command.

By default, this check is performed on

- All routing layers

To restrict the routing layers that are checked, use the `-ignore_pin_shape_layers` option to specify the layers to ignore. Specify the layers by using the layer names from the technology file.

- All cell instances of all cell types

To restrict the types of cells that are checked, use the `-ignore_pins_by_cell_type` option to specify one or more of the following values: antenna, chip, tap, cover, macro, soft_macro_io_fixed, io_pad, corner_pad, pad_filler, flip_chip_pad, flip_chip_driver, std, std_filler, double_height_std, triple_height_std, triple_more_height_std, and through_silicon_via. Cells whose `mask_layout_type` attribute matches a specified value are not checked.

If the set of floating pin shapes reported by this check contains overlapping or touching pin shapes that are connected to the same supply net, you can remove these pin shapes from the error set by using the `-ignore_touching_floating_pin_shape_errors` option.

For information about how electrical equivalence is determined, see “[Controlling the Tracing Behavior](#)” on page 11-27.

Defining a Strategy for the Dangling Pin Access Edge Check

To enable the dangling pin access edge check, use the `-dangling_pin_shape_edge_check` option with the `set_rail_integrity_layout_check_strategy` command.

By default, this check is performed on

- All routing layers

To restrict the routing layers that are checked, use the `-ignore_pin_shape_layers` option to specify the layers to ignore. Specify the layers by using the layer names from the technology file.

- All cell instances of all cell types

To restrict the types of cells that are checked, use the `-ignore_pins_by_cell_type` option to specify one or more of the following values: antenna, chip, tap, cover, macro, soft_macro_io_fixed, io_pad, corner_pad, pad_filler, flip_chip_pad, flip_chip_driver, std, std_filler, double_height_std, triple_height_std, triple_more_height_std, and through_silicon_via. Cells whose `mask_layout_type` attribute match a specified value are not checked.

Note that this check does not report errors on floating access edges. To check for these errors, enable the floating pin shapes check, as described in “[Defining a Strategy for the Floating Pin Shapes Check](#)” on page 11-21.

For information about how electrical equivalence is determined, see “[Controlling the Tracing Behavior](#)” on page 11-27.

Defining a Strategy for the Dangling Vias Check

To enable the dangling vias check, use the `-ideal_sourced_dangling_vias` option with the `set_rail_integrity_layout_check_strategy` command.

Note:

If a dangling via is also floating, you must fix the floating error first, and then check whether it is dangling.

For information about how to control the tracing behavior used for this check, see “[Controlling the Tracing Behavior](#)” on page 11-27.

Defining a Strategy for the Missing Vias Check

To enable the missing vias check, use the `-missing_via_rule` option with the `set_rail_integrity_layout_check_strategy` command. When you use this option you must specify each endpoint for the check using the following format:

layers objects

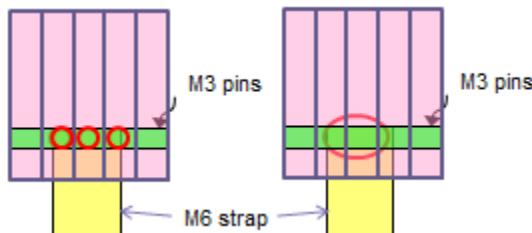
where the *layers* argument is one or more metal layers or the `ALL` keyword and the *objects* argument is a list of one or more of the following object types: `STRAP`, `RING`, `MACRO_CONN`, `STD_CONN`, `MACRO_PIN`, `STD_PIN`, `IO_PAD_PIN`, `FLIP_CHIP_DRIVER_PIN`, `FLIP_CHIP_PAD_PIN`, `ALL_WIRE`, `ALL_PIN`, or `ALL`.

By default, when performing the missing via check, the checking engine

- Identifies missing via candidates between overlapping metal shapes from the specified endpoints
To consider a missing via only when the distance between two neighboring vias along the direction of overlapping parallel wires is larger than a specified distance, use the `-max_via_distance_along_parallel_wires` option to specify the distance. Note that when you use this option, the engine saves the errors under the `MissViaMaxDist` category in the error browser.
- Ignores overlapping areas that are smaller than a minimum-sized via
To explicitly specify the maximum size of overlapping areas to ignore, use the `-ignore_small_overlapping_area` option.
To include these areas in the check, use the `-ignore_minimum_via_criterion` option. Note that using this option increases the number of errors, some of which could be false errors.
- Checks for well-aligned stacked vias
To allow staircase stacked vias, use the `-existing_stacked_via_check_mode via_enclosure` option.
To disable stacked via checking and check for missing vias only in adjacent metal routing layers, use the `-no_stack` option.
- Checks only perpendicular shapes
To control the relative direction of checked shapes, use the `-missing_via_net_shape_direction` option. You can specify `perpendicular`, `parallel`, or `both`.

- Considers via enclosure metal shapes as endpoints
To remove via enclosure metal shapes from consideration as endpoint shapes, use the `-ignore_via_enclosure_shape` option.
To allow the endpoint of a stacked via to be outside the overlapping area, use the `-allow_stacked_endpoint_vias_outside_of_overlapping_area` option.
- Considers each shape individually
To merge all shapes on the same layer associated with the specified shape types before performing the missing via check, use the `-merge_same_layer_shapes` option. When an error is detected, the command reports a single rectangle that represents the bounding box of the endpoint objects of the merged shapes.
For example, in [Figure 11-4](#) there are three pin shapes that overlap with the M6 strap. By default, the missing via check returns either three errors, one for each pin shape, or no errors, if the pin shapes are smaller than the minimum via size. With the `-merge_same_layer_shapes pin_shape` option, the pin shapes are merged before performing the check, and a single error is returned.

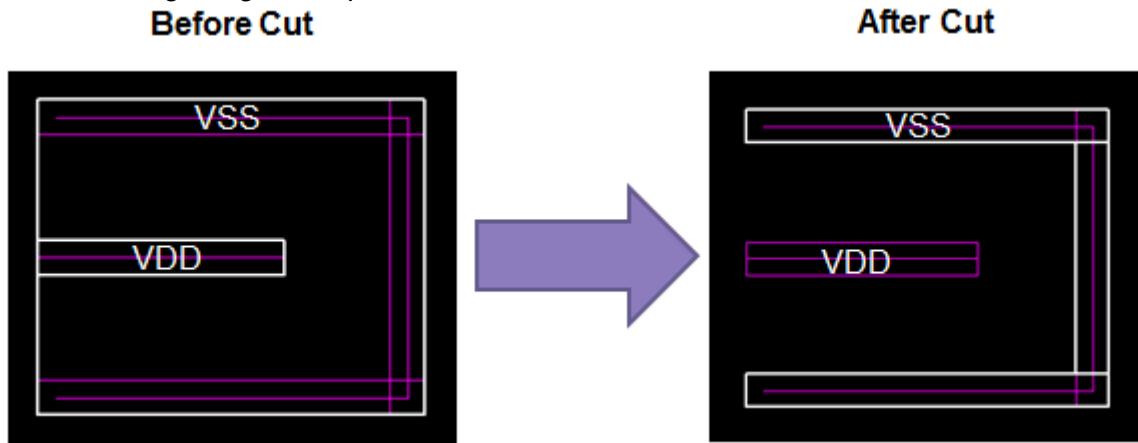
Figure 11-4 Merging Pin Shapes for Missing Via Checking



If you use the `-merge_same_layer_shapes` option with the `-missing_via_net_shape_direction both` option, the merged checking region might cover other shapes.

For example, [Figure 11-5](#) shows a U-shaped structure for the VSS net. When you use the `-merge_same_layer_shapes net_shape` option with the `-missing_via_net_shape_direction both` option, the command reports a rectangular error that covers the VDD net shape, as shown on the left side of [Figure 11-5](#). To avoid this situation, use the `-cut_merged_shape` option to break up the returned error into multiple rectangles that represent the pure overlapping area, as shown on the right side of [Figure 11-5](#).

Figure 11-5 Cutting Merged Shapes



- Considers existing vias only if they are fully enclosed in the overlapping area
To consider both fully and partially enclosed vias as existing vias, use the `-via_existence_check_partially_enclosed` option.
To merge all of the via cuts of a via array into one virtual via cut for enclosure checking, use the `-via_existence_check_merge_via_cut` option.
- Honors routing blockages and does not presume missing vias if the blockages overlap the area being considered for missing vias
To ignore all routing blockages on specific blockage layers, use the `-ignore_routing_blockage_layers` option. To ignore specific routing blockages, use the `-ignore_routing_blockages` option.
- Considers shapes belonging to other supply nets as blockages
To ignore the shapes belonging to other supply nets, use the `-ignore_other_pg_net_shapes` option.

Defining a Strategy for the Discontinuous Connections Check

To enable the discontinuous connections check, use the `-enable_discontinuous_connection_check` option with the `set_rail_integrity_layout_check_strategy` command.

By default, when performing this check, the checking engine

- Uses the minimum spacing for a layer as the gap threshold

To specify the maximum gap distance per layer, use the `-discontinuous_connection_threshold` option.

- Checks all routing layers

To ignore specific routing layers during checking, use the

`-ignore_discontinuous_connection_routing_layers` option to specify the routing layers to ignore. Specify the routing layers by using the Milkyway routing layer names, not the mask names.

- Checks both horizontal and vertical shapes

To check only shapes with a specific direction, use the

`-discontinuous_connection_directionality` option to specify the checked shape direction for each layer. You can specify `H` (horizontal), `V` (vertical), or `HV` (both).

Note:

The tool ignores the 45-degree paths.

- Checks all object types

Use the `-ignore_discontinuous_connection_object_types` option to specify which object types to ignore during discontinuous connection checking.

- Checks all shape types

Use the `-ignore_discontinuous_connection_shape_types` option to specify which shape types to ignore during discontinuous connection checking.

- Checks for other blocking nets within a gap

Use the `-ignore_other_shape_net_types` option to specify which net types to ignore during discontinuous connection checking.

Controlling the Tracing Behavior

During DRC or LVS checking, floating pin shapes and floating shapes are checked relative to the terminals of a physical block. In contrast, during rail integrity checking these checks are performed with taps or voltage source rail constraints. If there is no path from a shape to a tap, the engine flags it as an error.

By default, the IC Validator tool uses the `eeq_class` attribute in the FRAM views to model internal connectivity. If an instance has a CONN view and you specify the `-use_conn_view` option, the IC Validator tool uses the explicit internal physical connectivity in the CONN view, instead.

Use the `-pg_pin_floating_connection_check` option to check if any pin shapes are physically connected to an ideal voltage source. When you set this option to `minimal`, the tool identifies a pin as not floating when at least one pin shape of the pin that has the same `eeq_class` or `conn_view_eeq_class` value is not floating. When you set this option to `maximal`, the tool identifies a pin as not floating when all of its pin shapes that belong to the same `eeq_class` or `conn_view_eeq_class` value are not floating.

If you do not want the tool to use the `eeq_class` or explicit connectivity for tracing, you can

- Force all pin shapes to be considered as connected internally
 - To enable complete connectivity for FRAM views, use the `-override_eeq_class_to_interpret_as_same_class` option with the `set_rail_integrity_layout_check_strategy` command.
 - To enable complete connectivity for CONN views, use the `-override_conn_view_eeq_class_to_interpret_as_same_class` option with the `set_rail_integrity_layout_check_strategy` command.
- Force all pin shapes to be considered as unconnected internally
 - To disable connectivity for FRAM views, use the `-ignore_eeq_for_physical_connectivity_tracing` option with the `set_rail_integrity_layout_check_strategy` command.
 - To disable connectivity for CONN views, use the `-ignore_conn_view_eeq_for_physical_connectivity_tracing` option with the `set_rail_integrity_layout_check_strategy` command.

Filtering the Errors Based on Area

To report checking errors in a specified area only, you can

- Use the `-core`, `-polygon`, `-power_plan_regions`, `-voltage_areas`, or `-macros` option to specify the area within which to report errors.
- Use the `-exclude_macros` option to specify the area within which not to report errors.
- Use the `-area_edge_condition` option to specify the condition for including errors: the error boundary must be either completely within the specified area (`within`), within or touching the boundary of the specified area (`touching`), or within or overlapping the boundary of the specified area (`intersecting`). The default is `touching`.

Running the Rail Integrity Checking

To run the rail integrity checking, use the `verify_rail_integrity` command. When you run this command, you must use the `-integrity_layout_strategies` option to specify the rail integrity layout check strategies to use.

When you run the `verify_rail_integrity` command, PrimeRail generates the following files in the working directory:

- The PrimeRail script file

The script file is stored in a file named `verify_rail_integrity.tcl.timeStamp`. It contains the PrimeRail commands that were used to run the analysis. If an error occurs, you can modify the script file and use it in a subsequent run. For debugging purposes, you can also run the script in PrimeRail outside of the IC Compiler environment.

Note:

To generate the PrimeRail script file without actually running the analysis, use the `-script_only` option with the `verify_rail_integrity` command. You can modify and reuse this script file as needed.

- The generated result data

The integrity checking engine saves the generated results and log files in a directory named `RAIL_DATABASE/design_name/rail` and retrieves the data from this directory during verification.

- The PrimeRail log file

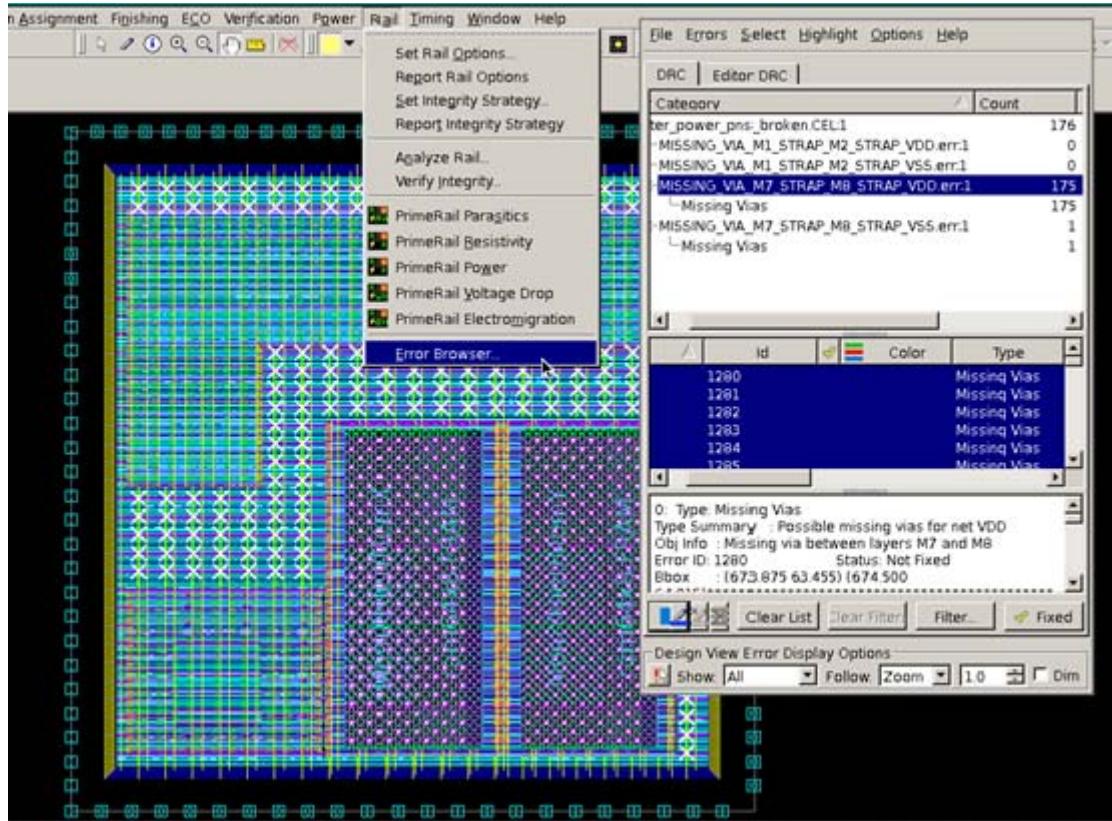
The log file is stored in a file named `verify_rail_integrity.log.timeStamp`.

PrimeRail also creates error views named `strategyName_supplyNetName` in the Milkyway design library. Multiple error views are generated per strategy as a function of the number of supply nets. The error view contains the locations of the issues found during rail integrity

checking. You can use the error browser to examine the errors in the GUI by choosing Rail > Error Browser.

[Figure 11-6](#) shows a generated error view displayed in the error browser.

Figure 11-6 Rail Integrity Checking Results in the Error Browser



For more information about the error browser, see “[Examining Routing and Verification Errors](#)” on page [A-94](#) and the “[Examining Routing and Verification Errors](#)” topic in IC Compiler Help.

Rail Integrity Checking Example

[Example 11-2](#) shows a script that performs In-Design rail integrity checking in IC Compiler. The script defines two rail integrity layout check strategies for the VDD and VSS supply nets: a strategy named MISSING_VIA_M1_STRAP_M2_STRAP that checks for missing vias between the M1 straps and M2 straps and a strategy named MISSING_VIA_M7_STRAP_M8_STRAP that checks for missing vias between the M7 straps and M8 straps. These strategy names become the prefixes for the error views that are created to report the results of the strategy.

Example 11-2 Rail Integrity Checking Script

```

#-----
# Specify the general rail analysis options
#-----
set_rail_options -user_defined_tap_file $tapfile \
    -output_dir pr_integrity_dir \
    -pr_exec_dir $PR_TCL_EXEC_PATH

#-----
# Remove all rail integrity layout check strategies
#-----
remove_rail_integrity_layout_check_strategy -all

#-----
# Define the rail integrity layout check strategies
#-----
set_rail_integrity_layout_check_strategy MISSING_VIA_M1_STRAP_M2_STRAP \
    -missing_via_rule { M1 STRAP M2 STRAP } -supply_nets [list VDD VSS]
set_rail_integrity_layout_check_strategy MISSING_VIA_M7_STRAP_M8_STRAP \
    -missing_via_rule { M7 STRAP M8 STRAP } -supply_nets [list VDD VSS]

#-----
# Report the rail integrity layout check strategies
#-----
report_rail_integrity_layout_check_strategy

#-----
# Perform rail integrity checking
#-----
verify_rail_integrity \
    -integrity_layout_strategies \
        {MISSING_VIA_M1_STRAP_M2_STRAP MISSING_VIA_M7_STRAP_M8_STRAP}

#-----
# Write rail strategies
#-----
write_rail_integrity_strategy

```

Performing Rail Integrity Checking in the GUI

To create or edit a rail integrity layout strategy and perform rail integrity checking in the GUI, choose Rail > Verify Rail Integrity, which opens the Verify Rail Integrity dialog box.

To create a new strategy, click New in the Verify Rail Integrity dialog box and configure the necessary settings in the Integrity Check-Strategy Editor window. To edit or delete an existing strategy, click Re-define or Delete in the Verify Rail Integrity dialog box. For information about the options in the Integrity Check -Strategy Editor, see “[Defining the Rail Integrity Layout Check Strategies](#)” on page 11-20.

To save the strategies and perform rail integrity checking on the current physical hierarchy of the open Milkyway cell, click OK or Apply.

Note:

The defined strategies are available only in the current session; they are not saved in the design database. To save a strategy for use in a subsequent session, use the `write_rail_integrity_strategy` command.

Figure 11-7 Verifying Rail Integrity in the GUI



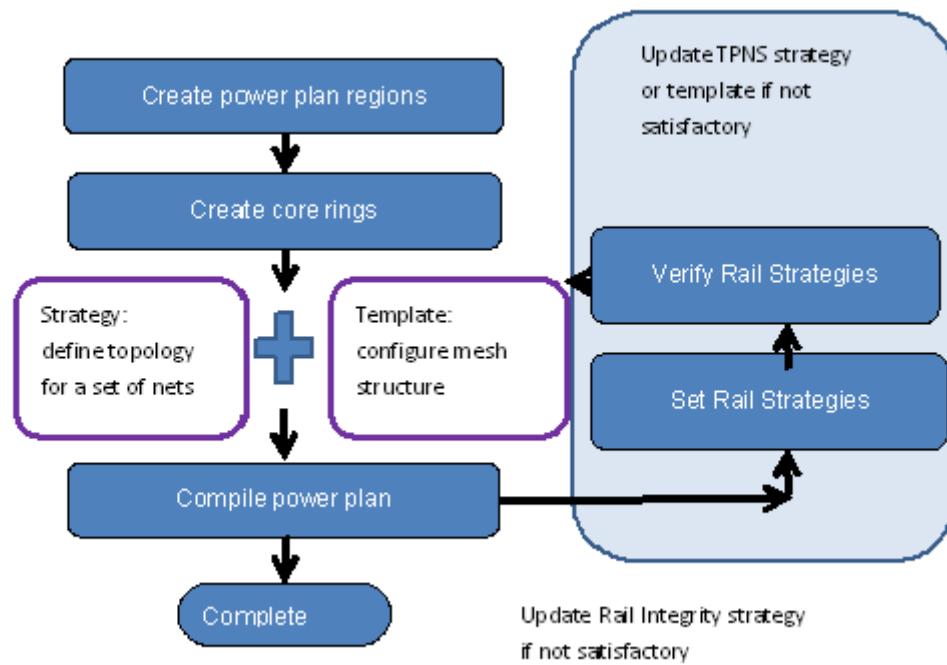
Integration With Template-Based Power Network Synthesis

When implementing a power network in the IC Compiler tool, one of the methods involves the use of template-based power network synthesis. This capability allows you to describe strategies and create common template components for different areas of a design to create strap and ring power structures at interactive speed. This involves constructing multiple strategies for portions of a power plan, such as rings or meshes, which configure templates by way of parameter settings, and then compiling the strategies to generate a physical implementation.

If you defined your power plan by using template-based power network synthesis to create power rings and power meshes, you can edit your strategies in the Integrity Check-Strategy Editor window. For example, you can define a strategy to check for missing vias on the generated structure. In-Design rail integrity checking is designed to work well with the IC Compiler template-based power network synthesis capability.

[Figure 11-6](#) shows the flow between template-based power network synthesis and In-Design rail integrity checking.

Figure 11-8 Interaction Between Template-Based Power Network Synthesis and In-Design Rail Integrity Checking



For more information about the template-based power network synthesis capability, see the power planning chapter in the *IC Compiler Design Planning User Guide*.

Performing Voltage Drop Analysis

Voltage drop analysis reports the voltage drop on a power net. Voltage drop can cause delay variations across the chip and can affect clock skew. You can perform voltage drop analysis in either static, dynamic, or dynamic vector-free mode, depending on the purpose of the analysis or the availability of the net switching information. You can perform static voltage drop analysis after the design is placed. You can perform dynamic voltage drop analysis (with or without vectors) after the design is routed.

To perform voltage drop analysis,

1. Set the general rail analysis options, including the analysis mode.

By default, the tool performs static voltage drop analysis. To perform dynamic voltage drop analysis, enable dynamic analysis mode by running the `set_rail_options` command with the `-analysis_mode dynamic` option. To perform vector-free dynamic

voltage drop analysis, enable dynamic vector-free analysis mode by running the `set_rail_options` command with the `-analysis_mode dynamic_vectorfree` option.

For details, see “[Setting the Rail Analysis Options](#)” on page 11-9.

2. Set the voltage drop analysis options.

For details, see “[Defining the Switching Activity Information](#)” on page 11-33, “[Defining the Analysis Scope](#)” on page 11-34, “[Enabling Effective Voltage Drop Reporting](#)” on page 11-34, and “[Reusing Rail Analysis Data](#)” on page 11-35.

3. Run the voltage drop analysis.

For details, see “[Running the Voltage Drop Analysis](#)” on page 11-36.

Defining the Switching Activity Information

By default, In-Design rail analysis uses the default toggle rate. Alternatively, In-Design rail analysis can get the switching activity information from a VCD file, a SAIF file, or a Tcl script.

Note:

if you are using the dynamic vector-free mode, you do not need to provide switching activity information.

To specify the switching activity file, use the `set_rail_options` command with the `-switching_activity` option, which has the following syntax:

```
-switching_activity {type file_name [strip_path]}
```

- For a VCD file, use one of the following type values:
 - For a VCD file generated from a gate-level simulation, use `VCD`.
 - For a VCD file generated from an RTL simulation, use `RTL_VCD`.
 - For a VCD file generated from a zero-delay simulation, use `ZERO_DELAY_VCD`.

By default, rail analysis uses all time values in the VCD file. To specify a time window, use the `-start_time` and `-end_time` options with the `set_rail_options` command.

The VCD file typically references a top-level design in a different hierarchy. Use the optional `strip_path` argument to specify the string to remove from the beginning of the object names.

- For a SAIF file, use `SAIF` as the type value.

The SAIF file typically references a top-level design in a different hierarchy. Use the optional `strip_path` argument to specify the string to remove from the beginning of the object names.

- For a Tcl file, use `User-defined` as the type value. The Tcl file can contain the following commands and variables:

Commands:

```
set_case_analysis  
set_switching_activity  
set_annotated_power
```

Variables:

```
power_default_toggle_rate  
power_default_static_probability  
power_default_toggle_rate_reference_clock
```

Defining the Analysis Scope

By default, the `analyze_rail` command performs rail analysis on the entire design, including all subblocks.

In the early stages of hierarchical design, when the design floorplans are divided into subblocks, top-level and block-level implementations might be done at the same time by different design groups. In the absence of subblocks, you must do a top-level rail analysis to check the robustness of the overall power network.

To restrict the rail analysis to the top-level power routing of the design, use the `-top_level_only true` option with the `set_rail_options` command. When you use this option, the `analyze_rail` command generates the script file and the necessary SBPF and SDC files only for the top-level design. No SBPF files are created for the subblocks.

Enabling Effective Voltage Drop Reporting

The effective voltage drop for a cell instance is the maximum difference between the voltage drop and voltage rise points at a given time step over the simulation time.

By default, the `analyze_rail` command does not report the effective voltage drops. To enable this reporting, use the `-report_effective_voltage_drop true` option with the `set_rail_options` command. By default, when the `analyze_rail` command reports the effective voltage drops, it computes the voltage differences for every time step during the simulation. To compute the voltage differences only for the timing windows in which the cell is in transition, use the `-timing_windows_only true` option with the `set_rail_options` command.

Reusing Rail Analysis Data

If you have already run rail analysis in the current session, you can reuse some or all of the previous results and setup information, which reduces the overall turnaround time.

To reuse the previously generated data, run the `set_rail_options -reuse` command before running the voltage drop analysis to specify which steps of the flow to reuse. You can reuse one or more of the following data items:

- PG extraction (`pg_extraction`)

The extraction on the supply nets is not performed. Instead, the SBPF file from the previous run is reused.

- Signal extraction (`signal_extraction`)

If you have already performed extraction on the design by using the `extract_rc` command, you can reuse the signal extraction information during In-Design rail analysis. This saves total runtime when rerunning In-Design rail analysis with only minor changes, such as reinserting taps.

The tool reuses the signal extraction results when possible. If the signal routing in a block has been modified since the extraction, the tool performs incremental extraction on that block. The tool uses the TLUPlus settings, SDC settings, reference cell modification date, and signal extraction results modification date to determine if the signal extraction results can be reused.

- Power analysis (`power`)

The power calculation performed by the PrimeRail `update_currents` command is not performed. Instead, the power report files created in the previous run are reused.

- Rail setup file (`setup_files`)

The rail setup file is not updated. Instead, the setup file from the previous run is reused.

- Rail setup variables (`setup_variables`)

The rail setup variables are not updated. Instead, the setup variables from the previous run are reused.

Running the Voltage Drop Analysis

To run the voltage drop analysis, use the `analyze_rail -voltage_drop` command (or choose Rail > Analyze Rail and select “Voltage Drop analysis” in the GUI).

In multivoltage designs, rail analysis involves multiple supply nets. To analyze an individual supply net or a group of supply nets, specify the nets as an argument to the `analyze_rail` command.

```
icc_shell> analyze_rail -voltage_drop {VDD VDDG VDDGS}  
icc_shell> analyze_rail -voltage_drop {VSS}
```

In power-gating designs, it is useful to analyze the entire power network of a power domain including the effects of power switches. The tool automatically runs combined voltage drop analysis on the primary and virtual power nets, along with the switches, as a single power network. To do the combined analysis, ensure that the on-state resistance values are specified for the power-switch cells. For more information about defining the on-state resistance values, see [“Library Requirements” on page 11-7](#).

When you run the `analyze_rail` command, the IC Compiler tool first creates setup and other data for In-Design rail analysis. In-Design rail analysis then loads the rail setup information and performs power analysis, PG extraction, and rail analysis. If you have already run rail analysis in the current session, you can reuse some or all of the previous data, which reduces the overall turnaround time. For information about reusing the previous data, see [“Reusing Rail Analysis Data” on page 11-35](#).

When voltage drop analysis is complete, the tool generates the following files in the working directory:

- A voltage drop violation report

The voltage drop violation report is stored in a file, named `rail_map_data/design_supplyNetName_err.txt`. It contains the voltage drop violations detected during voltage drop analysis.

- A voltage drop map for the specified supply nets

The map data is stored in a file named `rail_map_data/design_supplyNetName.map`.

The voltage drop map is a visual display of the colored range of voltage drop values overlaid on the physical supply nets. To view the voltage drop map in the GUI, choose Rail > PrimeRail Voltage Drop (or run the `read_rail_maps` command). For static analysis, the map displays average voltage drop values. For dynamic analysis, the map can display peak voltage drop values, average voltage drop values, or peak voltage rise values. For more information about viewing the voltage drop map, see the “Displaying the PrimeRail Voltage Drop Map” topic in IC Compiler Help.

- A parasitics map for the specified supply nets

The map data is stored in a file named `rail_map_data/design_supplyNetName.map`.

The parasitics map displays the parasitic resistance of a given supply net in the design. It shows the resistance for any given polygon of the net. To view the parasitics map in the GUI, choose Rail > PrimeRail Parasitics (or run the `read_rail_maps` command). For more information about viewing the parasitics map, see the “Displaying the PrimeRail Parasitics Map” topic in IC Compiler Help.

- A power map for the design

The power map data is stored in a file named `rail_map_data/design.map`.

The power map displays the power distribution for the design. To view the power map in the GUI, choose Rail > PrimeRail Voltage Drop (or run the `read_rail_maps` command). For more information about viewing the power map, see the “Displaying the PrimeRail Power Map” topic in IC Compiler Help.

- The PrimeRail setup and design input data

The PrimeRail setup and design input data (such as SDC and SBPF files) is stored in a directory named `synopsys_rail_setup`.

- The generated result data

In-Design rail analysis saves the generated analysis results and log files in a directory named `RAIL_DATABASE` and retrieves the data from this directory during analysis.

This directory contains the following subdirectories:

- `pgext`

This directory contains the supply net extraction results.

- `power`

This directory contains the power analysis results.

- `design_name/rail`

This directory contains the rail analysis results.

- The generated checking data

In-Design rail analysis saves the warning and error messages generated during the various analysis stages in a directory named `PR_CHECKING_DIR`.

This directory contains the following files:

- `lib.missing_CCS_power_current_waveform`

This file contains the names of the library cells that do not have CCS power current waveforms.

- mwdb.converting_pg_pin
This file contains the power and ground pins that are inferred from Milkyway but are not in the .db file.
 - mwdb.inferred_pg_pin
This file contains the cells with power and ground pins that are derived from Milkyway instead of the .db file.
- The PrimeRail script file

The PrimeRail script file is stored in a file, named `analyze_rail.tcl.timeStamp`. It contains the PrimeRail commands that were used to run the analysis. If an error occurs, you can modify the script file and use it in a subsequent run. For debugging purposes, you can also run the script in PrimeRail outside of the IC Compiler environment.

Some of the commands in the script file generated for a typical rail analysis include

```
open_mw_cel
set_extract_supply_parasitics_options
extract_supply_parasitics
update_currents
read_supply_parasitics
calculate_rail_voltage
report_rail_checks
```

Note:

To generate the PrimeRail script file without actually running the analysis, use the `-script_only` option with the `analyze_rail` command. You can modify and reuse this script file as needed.

- The PrimeRail log file

The PrimeRail log file is stored in a file named `analyze_rail.log.timeStamp`.

PrimeRail also creates an error view called `design_supplyNetName_vd` in the Milkyway design library. The error view contains 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 by choosing Rail > Error Browser. For more information about the error browser, see “[Examining Routing and Verification Errors](#)” on page A-94 and the “Examining Routing and Verification Errors” topic in IC Compiler Help.

By default, the tool does not report the effective voltage drop. To report the effective voltage drop, use the

Performing Electromigration Analysis

You can perform electromigration analysis at any implementation stage after placement. However, the recommended methodology is to perform electromigration analysis after detail routing, perform any necessary corrections, and then rerun electromigration analysis after chip finishing.

To perform electromigration analysis,

1. Set the general rail analysis options, including the analysis mode.

By default, the tool performs static electromigration analysis. To perform dynamic electromigration analysis, enable dynamic analysis mode by running the `set_rail_options` command with the `-analysis_mode dynamic` option.

For details, see “[Setting the Rail Analysis Options](#)” on page 11-9.

2. Set the electromigration analysis options.

For details, see “[Defining the Switching Activity Information](#)” on page 11-33, “[Defining the Analysis Scope](#)” on page 11-34, and “[Reusing Rail Analysis Data](#)” on page 11-35.

3. Define the electromigration rules.

For details, see the following section, “[Defining the Electromigration Rules](#).”

4. Run the electromigration analysis.

For details, see “[Running the Electromigration Analysis](#)” on page 11-41.

Defining the Electromigration Rules

Electromigration rules establish layer-by-layer current density limits for the layout geometries. The rules can establish limits on the average current, peak current, and RMS current. In-Design rail analysis uses the electromigration rules to perform electromigration analysis on the power supply network.

Ideally, you should use an Advanced Library Format (ALF) or TLUPlus file to define the electromigration rules, as it defines width- and temperature-dependent current limits.

However, if these are not available, In-Design rail analysis can use the `maxCurrDensity` attributes from the Milkyway technology file.

- To use an ALF file, read the file in one of the following ways:

- Read the ALF file directly.

To do this, use the `set_rail_options` command with the `-em_rule` option. Use the following syntax for the `-em_rule` option:

```
-em_rule {ALF ALF_file_name}
```

- Read the ALF file from the Milkyway design library.

To do this, load the ALF file into the design library by using the `set_mw_technology_file -alf` command. Do not use the `-em_rule` option with the `set_rail_options` command.

For example, to load the ALF file named `my_alf_file.alf` into the design library named `my_designlib`, use the following command:

```
icc_shell> set_mw_technology_file -alf my_alf_file.alf my_designlib
```

If the specified design library already has electromigration data, that information is completely replaced by the new information from the ALF file.

Note:

If the ALF file contains library information in addition to the electromigration rules, the `set_mw_technology_file` command does not check or save the additional library information because the command performs only limited syntax checking of the non-electromigration portions in the electromigration rule file.

- To use a TLUPlus file, read the file by using the `-em_rule` option with the `set_rail_options` command.

To do this, use the `set_rail_options` command with the `-em_rule` option. Use the following syntax for the `-em_rule` option:

```
-em_rule {ITF-EM TLUPlus_file_name layer_mapping_file}
```

You must specify the layer mapping file, which maps the metal and via layer names in the Milkyway technology file to the layer names in the ITF file. For information about this file, see “[Setting Up the TLUPlus Files](#)” on page 3-40.

- To use the `maxCurrDensity` attributes from the Milkyway technology file, use the `-em_rule {MW_TECH}` option with the `set_rail_options` command.

You must ensure that the `maxCurrDensity` attribute is defined in the `Layer` section for each metal and via layer. This attribute defines the threshold current density that the layer can safely sustain. This information is process-dependent. Consult with your process engineer or documentation to determine the appropriate values.

If you do not specify the `-em_rule` option, the tool uses the following order of precedence to determine the electromigration rules:

- The electromigration rules stored in the Milkyway design library from an ALF file
- The `maxCurrDensity` attributes from the Milkyway technology file

Running the Electromigration Analysis

To perform electromigration analysis, use the `analyze_rail -electromigration` command (or choose Rail > Analyze Rail and select “Electromigration analysis” in the GUI). By default, the `analyze_rail -electromigration` command performs static electromigration analysis.

In multivoltage designs, rail analysis involves multiple supply nets. To analyze an individual supply net or a group of supply nets, specify the nets as an argument to the `analyze_rail` command.

```
icc_shell> analyze_rail -electromigration {VDD VDDG VDDGS}  
icc_shell> analyze_rail -electromigration {VSS}
```

When you run the `analyze_rail` command, the IC Compiler tool first creates setup and other data for In-Design rail analysis. In-Design rail analysis then loads the rail setup information and performs power analysis, PG extraction, and rail analysis. If you have already run rail analysis in the current session, you can reuse some or all of the previous data, which reduces the overall turnaround time. For information about reusing the previous data, see [“Reusing Rail Analysis Data” on page 11-35](#).

When electromigration analysis is complete, the tool generates the following files in the working directory:

- An electromigration report

The electromigration violation report is stored in a file named `rail_map_data/design_supplyNetName_em_err.txt`. It contains the current violations detected during electromigration analysis.

- An electromigration map

The electromigration map data is stored in a file named `rail_map_data/design_supplyNetName.map`.

The electromigration map is a visual display of the colored range of electromigration values overlaid on the physical supply nets. To view the electromigration map in the GUI, choose Rail > PrimeRail Electromigration (or run the `read_rail_maps` command). For static analysis, the map displays average electromigration values. For dynamic analysis, the map can display average electromigration values, peak electromigration values, or root mean square electromigration values. For more information about viewing the electromigration map, see the “Displaying the PrimeRail Electromigration Map” topic in IC Compiler Help.

- The PrimeRail setup and design input data

The PrimeRail setup and design input data (such as SDC and SBPF files) is stored in a directory named `synopsys_rail_setup`.

- The generated result data

In-Design rail analysis saves the generated analysis results and log files in a directory named RAIL_DATABASE and retrieves the data from this directory during analysis.

This directory contains the following subdirectories:

- pgext

This directory contains the supply net extraction results.

- power

This directory contains the power analysis results.

- *design_name/rail*

This directory contains the rail analysis results.

- The generated checking data

In-Design rail analysis saves the warning and error messages generated during the various analysis stages in a directory named PR_CHECKING_DIR.

This directory contains the following files:

- em.missing_rules

This file contains information for the layers that do not have electromigration rules defined.

- lib.missing_CCS_power_current_waveform

This file contains the names of the library cells that do not have CCS power current waveforms.

- mwdb.converting_pg_pin

This file contains the power and ground pins that are inferred from Milkyway but are not in the .db file.

- mwdb.inferred_pg_pin

This file contains the cells with power and ground pins that are derived from Milkyway instead of the .db file.

- The PrimeRail script file

The PrimeRail script file is stored in a file named analyze_rail.tcl.*timeStamp*. It contains the PrimeRail commands that were used to run the analysis. If an error occurs, you can

modify the script file and use it in a subsequent run. For debugging purposes, you can also run the script in PrimeRail outside of the IC Compiler environment.

Note:

To generate the PrimeRail script file without actually running the analysis, use the `-script_only` option with the `analyze_rail` command. You can modify and reuse this script file as needed.

- The PrimeRail log file

The PrimeRail log file is stored in a file named `analyze_rail.log.timeStamp`.

PrimeRail also creates an error view named `design_supplyNetName_em` in the Milkyway design library. The error view contains the locations of the electromigration violations found during electromigration analysis. You can use the error browser to examine the errors in the GUI by choosing Rail > Error Browser. For more information about the error browser, see “[Examining Routing and Verification Errors](#)” on page A-94 and the “Examining Routing and Verification Errors” topic in IC Compiler Help.

Performing Inrush Current Analysis

Inrush current is the sum of switching and short-circuit current through a power-switch cell when it is on. Inrush current analysis provides a power-up sequence flow that controls the simultaneous switching noise (inrush current) due to turning on the power-switch cells. Power-switch cells should be configured such that the amplitude of the peak inrush current is no greater than the amplitude of the current caused by switching during normal operation.

To perform the most accurate inrush current analysis,

- Run the analysis after you run the `place_opt` command

You can perform inrush current analysis after inserting and connecting the power-switch cells during design planning. However, the standard cell capacitance information is less accurate than after placement.

- Provide a simulation-based Verilog Change Dump (VCD) file as input

You can perform vector-free inrush current analysis early in the design cycle when the VCD file is not available. However, vector-free analysis might show unrealistic scenarios and result in overdesign of the supply networks. Therefore, you should always run a VCD-based analysis on the final design.

To perform inrush current analysis,

1. Set the general rail analysis options, including the analysis mode.

Inrush current analysis is always a dynamic analysis. By default, when you run inrush current analysis, the tool also performs dynamic voltage drop analysis. To perform only inrush current analysis, enable inrush-only mode by running the `set_rail_options` command with the `-analysis_mode inrush_only` option.

Note:

If you do not change the analysis mode, inrush current analysis ignores the `set_rail_options -analysis_mode static` setting and performs the analysis in dynamic mode.

For details, see “[Setting the Rail Analysis Options](#)” on page 11-9.

2. Generate a PrimeRail configuration file.

For details, see “[Generating a PrimeRail Configuration File](#)” on page 11-44.

3. Run the inrush current analysis.

For details, see “[Running the Inrush Current Analysis](#)” on page 11-45.

Generating a PrimeRail Configuration File

In the PrimeRail configuration file, you specify

- The wake-up threshold (optional)

The wake-up threshold is the percentage of the supply voltage considered as the stable controlled voltage level after power-up in inrush analysis. The default is 0.01.

You specify this value by using the following syntax:

```
define WAKEUP_THRESHOLD float
```

The `WAKEUP_THRESHOLD` option is used to generate the power management control file.

- The name of the power-switch cell event file

You specify this file name by using the following syntax:

```
define SWITCH_CELL_EVENT_FILE file_name
```

You can specify one event file at a time. Multiple files are not supported.

To specify the PrimeRail configuration file, use the `set_rail_options -config_file` command.

Running the Inrush Current Analysis

To perform inrush current analysis based on the arrival time of the power-switch control pins, use the `analyze_rail -inrush` command (or choose Rail > Analyze Rail and select “In-rush current analysis” in the GUI). By default, the `analyze_rail -inrush` command performs both inrush current analysis and dynamic voltage drop analysis.

PrimeRail can perform inrush current analysis on both actual power nets and on virtual power nets controlled by the power-switch cells. To specify a virtual power net, you must specify the complete path to the virtual power net and its supply voltage, as shown in the following example:

```
icc_shell> analyze_rail -inrush {VDD {inst/VDDV 1.08}}
```

When you use the `-inrush` option, the `analyze_rail` command calculates the inrush current and the time required for the virtual power net to reach the same voltage value as the actual power net (the wake-up time) based on the event file.

After the inrush current analysis is complete, PrimeRail generates the following output in the PrimeRail run directory:

- Fast Signal Database (FSDB) file

The FSDB file is stored in a file named `inrush.fsdb`. It contains the following waveform information:

- Total rush current waveform for the switch cell subgroup
- Rush current waveform through each power-switch cell
- The voltage waveform for the virtual power and ground net

You can view the FSDB file in a waveform viewer, such as nWave.

- A voltage drop map and a parasitics map for the specified power and ground nets

The map data is stored in a file named `rail_map_data/design_supplyNetName.map`.

The voltage drop map is a visual display of the colored range of voltage drop values overlaid on the physical supply nets. To view the voltage drop map in the GUI, choose Rail > PrimeRail Voltage Drop (or run the `read_rail_maps` command). The voltage drop map can display peak voltage drop values, average voltage drop values, or peak voltage rise values. For more information about viewing the voltage drop map, see the “Displaying the PrimeRail Voltage Drop Map” topic in IC Compiler Help.

The parasitics map displays the parasitic resistors of a given power and ground net in the design. It shows the resistance for any given polygon of the net. To view the parasitics map in the GUI, choose Rail > PrimeRail Parasitics (or run the `read_rail_maps` command). For more information about viewing the parasitics map, see the “Displaying the PrimeRail Parasitics Map” topic in IC Compiler Help.

- The PrimeRail setup data

The PrimeRail setup and design input data (such as SDC and SBPF files) is stored in a directory named `synopsys_rail_setup`.

- The PrimeRail script file

The PrimeRail script file is stored in a file named `analyze_rail.tcl.timeStamp`. It contains the PrimeRail commands that were used to run the analysis. If an error occurs, you can modify the script file and use it in a subsequent run. For debugging purposes, you can also run the script in PrimeRail outside of the IC Compiler environment.

Note:

To generate the PrimeRail script file without actually running the analysis, use the `-script_only` option with the `analyze_rail` command. You can modify and reuse this script file as needed.

- The PrimeRail log file

The PrimeRail log file is stored in a file named `analyze_rail.log.timeStamp`.

The log file includes the following report, which shows the number of power-switch cells (called power management cells in the report), the total off-state leakage current, the total on-state leakage current, the wake-up time, and the peak inrush current for each switching group:

```
Power Management Analysis Report

Controlled voltage domain 0: VDDGS
 273 power management cell channels are used to control 4211 std
  cells.
  Total OFF leakage current from PM cells is 19.386e-3 (uA).
  Total ON leakage current from std cells is 6.153 (uA).
 0% (989.831987) 10% 20% 30% 40% 50% (2.991832e3) 60% 70% 80% 90%
  (4.724832e3)
  Stable PG net voltage 1.069 (V) is reached at wakeup time 6.33e3
    [start at 525.832 duration 5.804e3] (ps).
  Peak current 23.618e-3 (A) is required at 1.645e3 (ps).
```

You can use the physical power information to refine the number and sizing of the power-switch cells. You can use the wake-up time and peak inrush current information to validate the switch control assumptions.

Performing Minimum Path Resistance Analysis

Minimum path resistance analysis calculates path resistances from each point in the power and ground network to the tap point on any of the target nets that are reachable along the least resistive path.

To perform minimum path resistance analysis,

1. Set the general rail analysis options.

For minimum path resistance analysis, you should enable the use of the top-level power and ground pins as ideal voltage sources by using the `set_rail_options` command to set the `-use_pins_as_pads` option to `true`.

For details about setting the rail analysis options, see “[Setting the Rail Analysis Options](#)” on page 11-9.

2. Run the minimum path resistance analysis.

For details, see “[Running the Minimum Path Resistance Analysis](#)” on page 11-47.

Running the Minimum Path Resistance Analysis

To perform minimum path resistance analysis, use the `analyze_rail -min_path_resistance` command (or choose Rail > Analyze Rail and select “Min Path Resistance analysis” in the GUI).

After the minimum path resistance analysis is complete, PrimeRail generates the following output in the PrimeRail run directory:

- A minimum path resistance map

The map data is stored in a file named `rail_map_data/design_supplyNetName.map`.

The minimum path resistance map is a visual display of the colored range of minimum path resistance values overlaid on the physical supply nets. To view the minimum path resistance map in the GUI, choose Rail > PrimeRail Min Path Resistance (or run the `read_rail_maps` command).

- The PrimeRail setup data

The PrimeRail setup and design input data (such as SDC and SBPF files) is stored in a directory named `synopsys_rail_setup`.

- The PrimeRail script file

The PrimeRail script file is stored in a file named `analyze_rail.tcl.timeStamp`. It contains the PrimeRail commands that were used to run the analysis. If an error occurs, you can

modify the script file and use it in a subsequent run. For debugging purposes, you can also run the script in PrimeRail outside of the IC Compiler environment.

Note:

To generate the PrimeRail script file without actually running the analysis, use the `-script_only` option with the `analyze_rail` command. You can modify and reuse this script file as needed.

- The PrimeRail log file

The PrimeRail log file is stored in a file named `analyze_rail.log.timeStamp`.

12

Multivoltage Design Flow

This chapter describes the multivoltage design flow in the IC Compiler tool. It contains the following sections:

- Multivoltage Designs
- High-Level Design Flow for Multivoltage Designs
- UPF Flow for Multivoltage Designs
- Golden UPF Flow
- Implementing Power Switches
- Connecting Power and Ground Nets
- Specifying the Operating Voltage
- Defining and Using Voltage Areas
- Power Management Cells and UPF Strategies
- N-Well and P-Well Bias
- Example IC Compiler Script for the UPF Flow
- Voltage-Area-Aware Capabilities
- Placing and Optimizing the Design
- Clock Tree Synthesis in Power-Down Domains

- Multivoltage-Specific Queries, Checks, and Reports
- Hierarchical UPF Flow

Multivoltage Designs

The IC Compiler tool supports multivoltage, multisupply, and mixed multivoltage-multisupply designs. In multivoltage designs, the subdesign instances, or blocks, operate at different voltages. In multisupply designs, the block voltages are the same, but the blocks can be powered on and off independently. (In this user guide, unless otherwise noted, the term *multivoltage* includes multisupply and mixed multisupply-multivoltage designs).

You use the IEEE 1801 Unified Power Format (UPF) to specify the power supply intent for your design. UPF supports the ability to specify the power intent for your design early in the design process. The IC Compiler tool supports the use of UPF for the entire design flow. For more details about the low-power flow and the various Synopsys tools that support UPF, see the *Synopsys Multivoltage Flow User Guide*.

Multivoltage designs typically make use of the following features:

- Power domains – groupings of logic hierarchies that share the same power supplies
 - Voltage areas – physical placement areas for the cells of a power domain
 - Power management cells – level shifters and isolation cells used to connect the drive and load pins across different power domains
-

Power Domains

Power consumption can be reduced in a multivoltage design by making use of power domains that are independently powered up and down, including domains that are defined to have always-on relationships relative to each other.

Specifically, a power domain is defined as a grouping of one or more logical hierarchies in a design block that share the following properties:

- Primary voltage states or voltage range (specifically, the same operating voltage)
- Power net hookup requirements
- Power-down control and acknowledge signals (if any)
- Power-switching style
- Process, voltage, and temperature (PVT) operating condition values (all cells of the power domain except level shifters)
- Same set or subset of nonlinear delay model (NLDM) target libraries
- Same physical voltage area into which the cells of these logic hierarchies are placed

You specify all the power information for your design in a separate file, the UPF file. You read this UPF file into the IC Compiler tool, along with your design. For more details, see “[UPF Flow for Multivoltage Designs](#)” on page 12-8.

Voltage Areas

A voltage area is the physical placement area of a logic block comprising one or more logic hierarchies operating under a single voltage level. Except for level-shifter cells, all cells in the hierarchies associated with a voltage area operate at the same process, voltage, and temperature (PVT) operating condition values.

You can create voltage areas either during floorplanning as described in the *IC Compiler Design Planning User Guide* or by using the `create_voltage_area` command. Voltage areas must be defined before running the `psynopt` or `place_opt` command.

Note:

If you use the floorplanning method to define voltage areas, the floorplan must contain a default voltage area for the top-level cells and at least one other voltage area. If you create the voltage areas directly by using the `create_voltage_area` command, the tool automatically derives a default voltage area for the top-level leaf cells and level shifters that do not belong to any block.

Physically nested voltage areas are supported, but intersecting voltage areas are not supported. The recommended way of creating physically nested voltage areas is to have one voltage area completely enclose the other.

For more information, see “[Defining and Using Voltage Areas](#)” on page 12-14 and “[Handling Nested Voltage Areas](#)” on page 12-19.

Associating Power Domains and Voltage Areas

There must be an exact one-to-one relationship between logical power domains and physical voltage areas. The Design Compiler and IC Compiler tools can automatically align the logic hierarchies of the power domains with their voltage areas if you provide the corresponding power domain name as an argument in each `create_voltage_area` command. In this case, the power domain name and the voltage area name are identical.

If you do not explicitly specify the power domain name when you use the `create_voltage_area` command, you are responsible for ensuring that the logic hierarchies are correctly aligned, as well as correctly associated with the appropriate operating conditions.

Level-Shifter and Isolation Cells

Level-shifter cells operate across voltage differences, connecting drive and load pins of cells belonging to different power domains (and placed in different voltage areas). The basic function of a level shifter is to connect the power domains by stepping the voltage up or down as needed. The IC Compiler tool supports both simple buffer-type level shifters and enable-type level shifters, which can perform both level shifting and isolation. Enable-type level shifters are used when the power domains must be powered on and off independently. A design can contain both types of level shifters.

By using the `insert_mv_cells` command, you can explicitly insert buffer-type level shifters into an unmapped or mapped netlist at any point in the design flow before placing the cells of the design. In an *unmapped* design, buffer-type level shifters are automatically inserted as needed when you run the `compile_ultra` or `compile` command. If you intend to explicitly insert level shifters, you should run the command as early in the flow as possible.

In UPF mode, you can insert enable-type level shifters using the `insert_mv_cells` command. Usually these level shifters are instantiated at the RTL level or represented by generic library (GTECH) cells, before the unmapped design is compiled. During logic synthesis, existing isolation cells are sometimes replaced with enable-type level shifters. (Since enable-type level shifters are buffers, they can also be used as isolation cells.)

You use isolation cells to isolate the input and output signals of the shut-down power domains. These cells can be instantiated at the RTL level of the design description, or you can run the `insert_mv_cells` command to insert them directly. (Note that isolation cells are sometimes replaced by enable-type level shifters during logic synthesis.)

The IC Compiler tool places level shifters and isolation cells near the voltage area boundaries, usually at the top level of the design. However, if the port voltages are appropriately specified, these cells can be placed inside the voltage areas. For some designs, there are advantages to placing these cells inside the voltage areas.

For more information, see “[Power Management Cells and UPF Strategies](#)” on page 12-22.

Physical Implementation

Use the IC Compiler `place_opt` command to perform the physical implementation phase of the multivoltage design flow. During this phase, the gate-level designs are read into memory along with the floorplan. The multivoltage design steps are

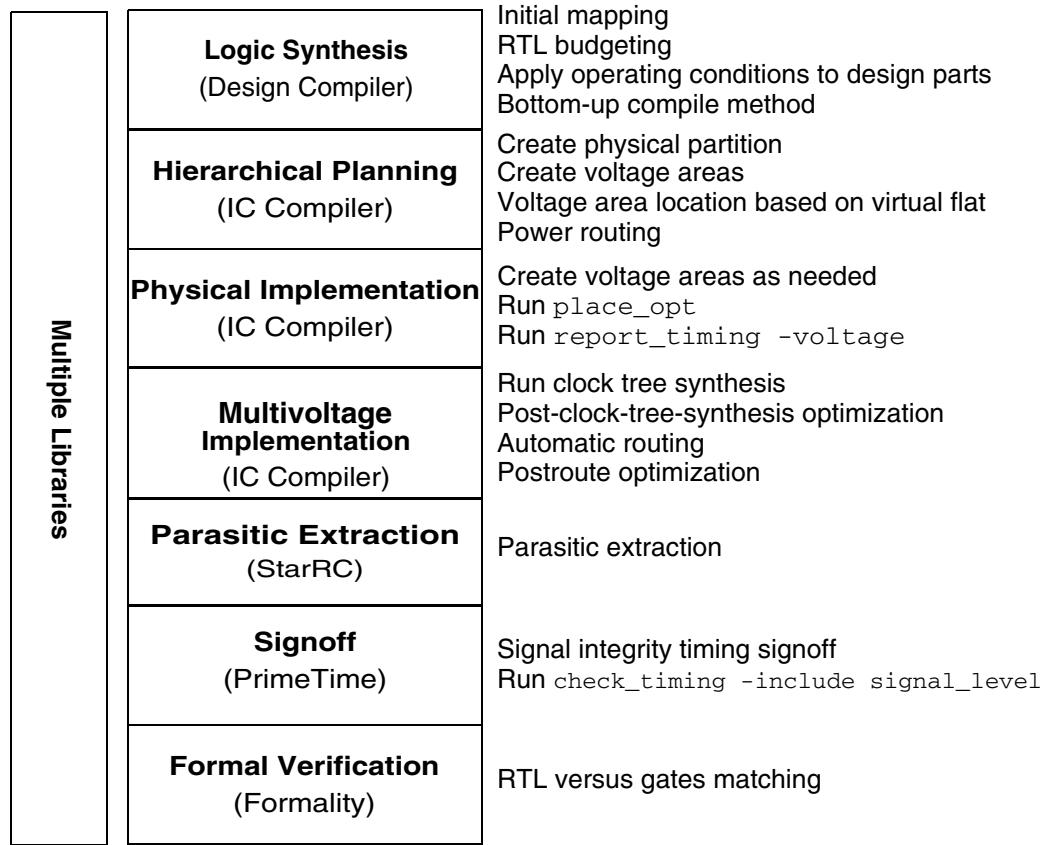
1. (Optional) Insert buffer-type level shifters if they were not inserted during the logic synthesis phase of the flow.
2. Transfer the power domain objects into the Milkyway database (Design Compiler to IC Compiler flow) or create power domains and power domain objects in the IC Compiler tool (third-party synthesized design to IC Compiler flow).

3. Create voltage areas and associate or align the logic hierarchies, and write the information into the Milkyway database.
4. (Optional) Create power wells within given voltage areas as always-on placement sites for single-power standard cells.
5. Run the `place_opt` command to perform cell placement in the corresponding voltage areas of the power domains and optimize the design. This includes always-on optimization of the paths that must receive power when shut-down power domains are powered down.

High-Level Design Flow for Multivoltage Designs

The IC Compiler tool support for multivoltage designs is part of a larger design flow involving several Synopsys tools. Support for multivoltage designs is automatically enabled across the entire Galaxy Design platform. [Figure 12-1](#) shows the Galaxy Design platform high-level flow for multivoltage designs. This is a complete RTL-to-signoff design flow. To understand how to use the individual tools of the design flow, consult the appropriate user guides.

Figure 12-1 Galaxy Design Platform High-Level Multivoltage Design Flow



In typical design flows, the multivoltage power intent of the design is specified by a UPF file at the synthesis stage using the Power Compiler tool. In that case, the Power Compiler tool writes out a UPF file that can be used as the power intent specification in the IC Compiler tool. The only additional usage of UPF in the IC Compiler tool might be to create and map power switches. You can then perform physical implementation of the power switches in the IC Compiler tool by placing arrays or rings of power switch cells in the design and connecting them to the power supply network.

If you did not use UPF commands in logic synthesis, you can specify the power intent of the design in the IC Compiler tool by using the full range of UPF commands to create power domains, supply nets, supply ports, isolation strategies, power state tables, and so on. For information about using UPF commands to perform these tasks, see the *Synopsys Multivoltage Flow User Guide*.

UPF Flow for Multivoltage Designs

Before you run the UPF flow, you must ensure that the target libraries comply with the power and ground pin Liberty library syntax. For information about the power and ground pin library syntax, see the advanced low-power modeling information in the *Library Compiler Timing, Signal Integrity, and Power Modeling User Guide*.

The steps involved in the UPF flow are

1. Read the multivoltage design.

The design being read into the IC Compiler tool should already contain special cells such as the retention registers, isolation cells, and level-shifter cells. For more details, see “[Reading the Design](#)” on page 3-12.

2. Read the UPF file.

Use the UPF commands to specify the power intent or power constraints for your multivoltage design in a UPF file, and use the `load_upf` command to read the UPF file into the IC Compiler tool.

3. Create and map the power switches.

Use the `map_power_switch` command to map each power switch to the logic library cell used to implement the switch. Use the `create_power_switch_ring` command to create a ring of power-switch cells around a voltage area, macro cell, or polygon. Use the `create_power_switch_array` command to instantiate the power-switch cells in the floorplan.

4. Create the logical power and ground connections.

After you read in the design, you must create logical connections between the power and ground nets and the power, ground, and tie-off pins of the cells in your design. Use the `derive_pg_connection` command to create these connections. Use the `-create_net` option to create the missing power nets.

For more information, see “[Creating Logical Power and Ground Connections](#)” on page 3-26 and the *IC Compiler Design Planning User Guide*.

5. Set the operating conditions on the top level of the design hierarchy, and set the power and ground voltage values.

Use the `set_operating_conditions` command to set the operating conditions and the `set_voltage` command to set the power and ground voltage values.

6. Check for multivoltage-specific violations.

Use the `check_mv_design` and `analyze_mv_design` commands to check for multivoltage-specific violations in the design. The `-verbose` option gives a detailed report of the violations.

7. Perform placement, clock tree synthesis and routing.

If you run the `place_opt`, `clock_opt`, and `route_opt` commands, the tool internally runs the `derive_pg_connection` command and updates the log file with a summary of the power and ground connections. If you use any optimization commands other than these core commands, run the `derive_pg_connection` command to complete the power and ground connections. Use the `check_mv_design -power_nets` command to check for mismatches in the power and ground pin connections.

8. Save the UPF file.

Use the `save_upf` command to write out a UPF file. Compared to the UPF file that you read into the tool, the UPF file written by the IC Compiler tool contains the following additional information:

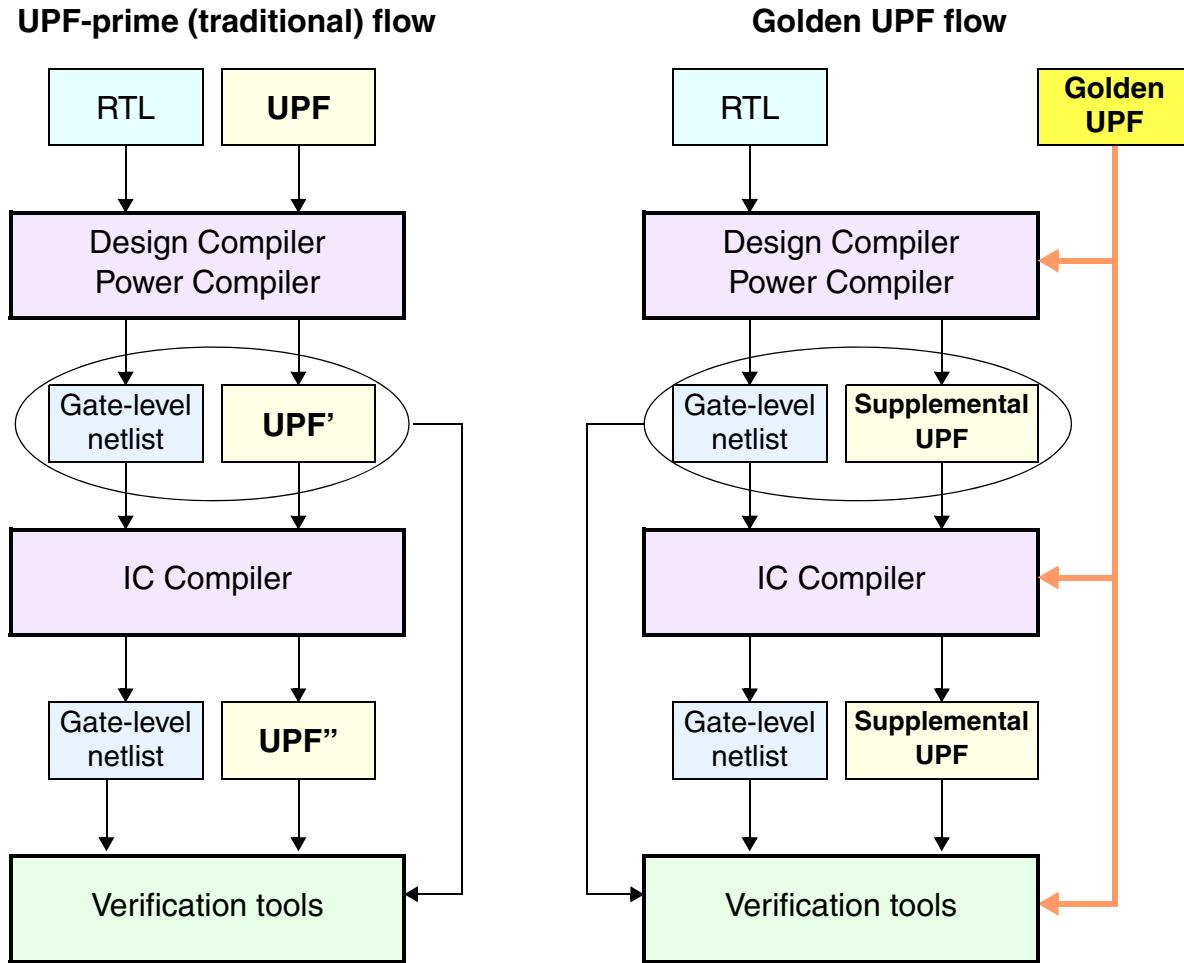
- A comment on the first line of the file indicating that it was generated by the IC Compiler tool.
- Explicit power connections to special cells such as level shifters and isolation cells.
- Any additional UPF commands that you specified at the command prompt during the IC Compiler session.

Golden UPF Flow

The golden UPF flow is an optional method of maintaining the UPF multivoltage power intent of the design. It uses the original “golden” UPF file throughout the synthesis, physical implementation, and verification steps, along with supplemental UPF files generated by the Design Compiler and IC Compiler tools.

[Figure 12-2](#) compares the traditional UPF flow with the golden UPF flow.

Figure 12-2 UPF-Prime (Traditional) and Golden UPF Flows



The golden UPF flow maintains and uses the same, original “golden” UPF file throughout the flow. The Design Compiler and IC Compiler tools write power intent changes into a separate “supplemental” UPF file. Downstream tools and verification tools use a combination of the golden UPF file and the supplemental UPF file, instead of a single UPF’ or UPF” file.

The golden UPF flow offers the following advantages:

- The golden UPF file remains unchanged throughout the flow, which keeps the form, structure, comment lines, and wildcard naming used in the UPF file as originally written.
- You can use tool-specific conditional statements to perform different tasks in different tools. Such statements are lost in the traditional UPF-prime flow.

- Changes to the power intent are easily tracked in the supplemental UPF file.
- You can optionally use the Verilog netlist to store all PG connectivity information, making `connect_supply_net` commands unnecessary in the UPF files. This can significantly simplify and reduce the overall size of the UPF files.

To use the golden UPF flow, you must enable it by setting the following variable:

```
icc_shell> set_app_var enable_golden_upf true
```

After you enable this mode, to execute any UPF commands other than query commands, you must put the commands into a script and execute them with the `load_upf` command. You cannot execute them individually on the command line or with the `source` command.

For more information about using the golden UPF mode, see SolvNet article 1412864, “Golden UPF Flow Application Note.”

Implementing Power Switches

A block that can be powered down must receive its power through a power-switching network, consisting of a large number of transistors with source-to-drain connections between the power supply rail and the power pins of the cells, or between the ground rail and the ground pins of the cells. The network, when switched on, connects the power to the logic gates in the block. When switched off, the power supply is effectively disconnected from the logic gates in the block.

This is the general procedure for implementing power switches in the IC Compiler tool:

1. If a power switch has not already been defined for a power domain in the UPF file, use the `create_power_switch` command to define it. This command specifies the power domain name, the switch input and output supply port names, the switch control port that invokes power-down, and the on state associated with the power domain.
The `create_power_switch` command only defines the general behavior of the power switch for a power domain. It does not actually place any switch cells in the physical chip layout.
2. Use the `create_power_switch_array` command to place a uniform array of switch cells throughout a specified area or the `create_power_switch_ring` command to place a ring of switch cells around a specified area. Each type of switch cell configuration, array-type or ring-type, has its advantages and disadvantages. For details, see “Creating Power-Switch Arrays and Rings” in the *IC Compiler Design Planning User Guide*.
3. Use the `connect_power_switch` command to connect the control ports of the switch cells to the power-down control logic. You can connect the switch cells in a daisy-chain, high-fanout, or fishbone configuration. For details, see “Connecting Power-Switches” in the *IC Compiler Design Planning User Guide*.

Several IC Compiler commands let you control the distribution of switch cells within or around the voltage area, to explore different switch configurations, and to optimize the switch implementation for resource usage versus IR drop. For details, see “Creating Power-Switch Arrays and Rings” in the *IC Compiler Design Planning User Guide*.

Connecting Power and Ground Nets

To connect the power and ground pins defined in your power domain to the appropriate power and ground supply ports of your design, use the `derive_pg_connection` command. By default, this command works only on the unconnected power and ground nets. Use the `-reconnect` option to force the command to work on the entire power network by reconnecting the already-connected power and ground pins as well as connecting the unconnected pins.

Use the `derive_pg_connection` command before you use any optimization commands. This command appropriately connects the power and ground pins defined in the design hierarchy based on the power and ground details defined in the power domains in the UPF file. This command can also create the power and ground connections for the physical-only cells.

The automatic connection behavior for power pins is as follows:

- For a power pin with an exception connection defined, the power net specified in the exception is used.
- For cells with a single power pin that are used as always-on cells, the backup power net is used.
- For cells with a single power pin that are *not* used as always-on cells, the primary power net is used.
- For multithreshold-CMOS switch cells with multiple power pins, the power pins are connected to the power domain power nets of the same type—that is, primary power pin to primary power net, backup power pin to backup power net, and internal power pin to internal power net.
- In the shut-down power domains of a multithreshold-CMOS design, the primary power pins of the standard cells are connected to the internal power net of the shut-down domain. The primary power net of the shut-down power domain is connected only to the primary pin of the switch cell.
- For boundary cells with multiple power pins (level shifters and isolation cells), the power pins are connected to the power nets of the related power domains—that is, the load pin of the boundary cell is connected to the power net of the driver’s power domain, and the drive pin of the boundary cell is connected to the power net of the load’s power domain.

- For cells with multiple power pins that are neither nor multithreshold-CMOS switch cells nor boundary cells (for example, multirail always-on cells, retention registers, and macro cells), the power pins are connected to the appropriate power nets—that is, the primary and backup power pins are connected to the primary and backup power nets, respectively.

Note:

The rules applicable to the automatic connection of ground pins are similar to those for the automatic connection of power pins.

To verify the power connections, use the `check_mv_design -power_nets` command.

Explicitly Identifying Power and Ground Nets

In a UPF black box flow, some supply or ground nets might be used in subblocks but not at the top level. By explicitly identifying these nets as power supply or ground nets, the IC Compiler tool has the information it needs to derive the newly created nets at the top level, and the power type information is also honored by other multivoltage commands such as `check_mv_design`.

To identify specific nets as power and ground nets, set the following variables to the names of the nets:

```
power_nets_for_upf_supply  
ground_nets_for_upf_supply
```

For example, to define V1 and V2 as power supply (not ground) nets, use the following command:

```
icc_shell> set_app_var power_nets_for_upf_supply {V1 V2}
```

It is not necessary to use this variable to identify UPF supply nets that already have real usage in the design, for example, when they are used as the primary, isolation, or retention power supply, or are explicitly connected to power pins of leaf cells. In these cases, you can derive the power type for these supply nets and later implement them as real power nets by using the `derive_pg_connection -create_nets` command.

Specifying the Operating Voltage

Use the `set_operating_conditions` command to set the operating conditions for the top level of your multivoltage design.

Use the `set_voltage` command to specify the minimum and maximum operating voltage for all the supply nets. All supply nets, including the ground net, must be assigned an operating voltage value.

Use the `set_voltage` command to set the operating voltage on the internal power and ground (PG) pins of the fine-grained switch cells. If you do not set the voltage on an internal PG pin of a fine-grained switch cell, the value of the `voltage_name` attribute of the PG pin is used as the operating voltage.

The operating voltage that you have already set cannot be removed. However, you can override the existing settings by using the `set_voltage` command again.

Use the `check_mv_design -power_nets` command before optimizing the design, to ensure that operating voltages are defined for all the supply nets. For more details, see [“Multivoltage-Specific Checks” on page 12-43](#).

Defining and Using Voltage Areas

A voltage area is a placement area for one or more logical partitions operating at the same voltage. The cells of the logical partition are associated with the partition’s voltage area and are constrained to placement within that area.

Use the `create_voltage_area` command to create voltage areas and align the hierarchies, as described in the *IC Compiler Design Planning User Guide*.

You can define voltage areas at any level of a logic hierarchy. A voltage area can be a single rectangular or rectilinear shape or can consist of multiple disjoint rectilinear and rectangular shapes.

Physically nested voltage areas are supported, provided the inner voltage is completely contained within the outer voltage area. These nested voltage areas correspond to logically nested power domains. Voltage areas that would overlap physically have to be resolved into nonoverlapping, abutted rectangular or rectilinear subareas. How you construct voltage areas in this case is discussed in [“Handling Nested Voltage Areas” on page 12-19](#).

Voltage areas can be explicitly associated with existing power domains, or they can be created without specifying any explicit association with the power domains. They can also be created when no power domains have been defined for the design.

The logic hierarchies are assigned to specific voltage areas when you create or update these voltage areas by specifying a cell list in the `create_voltage_area` command. You must ensure that the hierarchies are aligned correctly. The tool checks the operating condition voltage of each module against the voltage area specification to ensure that all modules inside the voltage area operate at the correct voltage.

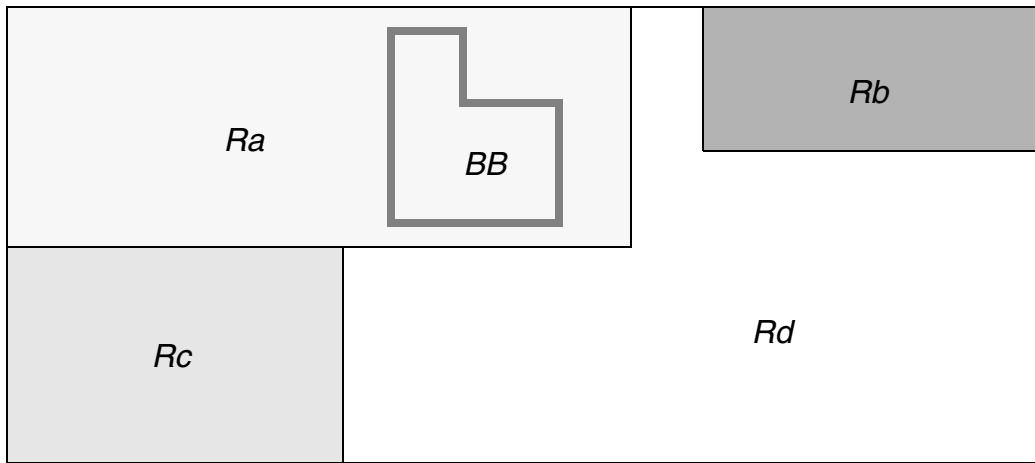
For designs with power domains, you can ensure cell alignment by specifying the name of the associated power domain when you run the `create_voltage_area` command. Use the `-power_domain` option to specify the name. In this case, you do not specify a voltage area name or the list of cells in the `create_voltage_area` command because the cell list is provided in the power domain definition.

Level shifters and isolation cells can have site heights that differ from the standard cell site height by either integer multiples (homogeneous sites) or noninteger multiples (heterogeneous sites). The IC Compiler tool supports the placement of both types of sites.

Using heterogeneous sites for certain level shifters and isolation cells can help improve utilization. However, if the design has overlapping rows, set the `physopt_check_site_array_overlap` variable to `false` before running placement and legalization.

[Figure 12-3](#) shows a multivoltage design divided into several voltage areas.

Figure 12-3 Multivoltage Design With Several Voltage Areas



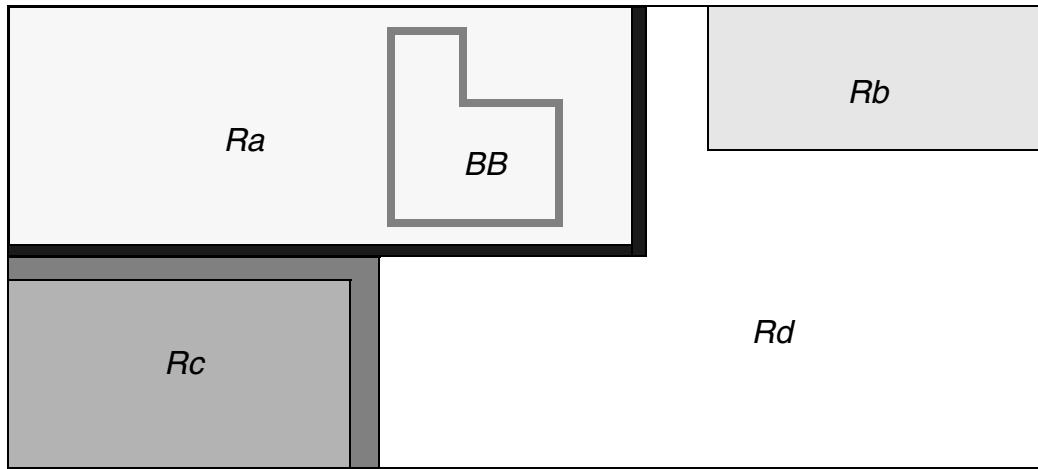
In this design, the floorplan has four voltage areas: Ra, Rb, Rc, and Rd. The default voltage area is Rd. Each voltage area has a corresponding logical partition, A, B, C, and D, that aligns with its respective voltage area. A hard bound can be defined inside a voltage area but not across voltage areas. In this example, a hard bound, BB, is defined inside voltage area Ra.

Special always-on site rows can be defined within a given voltage area. The standard cells placed in these site rows serve as the always-on cells of the shut-down power domain associated with the given voltage area.

Optionally, guard bands can be specified for some or all voltage areas to prevent shorts. Guard bands define hard keepout margins surrounding the voltage areas in which no cells, including level shifters and isolation cells, can be placed. Guard bands can be defined when creating voltage areas automatically or when updating the voltage areas.

[Figure 12-4](#) shows a floorplan similar to [Figure 12-3](#) but with guard bands protecting voltage areas Ra and Rc.

Figure 12-4 Multivoltage Design With Several Voltage Areas and Guard Bands



Creating Voltage Areas

You create voltage areas by using the `create_voltage_area` command. With this command you specify the exact coordinates of the voltage area.

If the design contains power domains, you can use the `-power_domain` option with the corresponding power domain name in the `create_voltage_area` command. Then the power domain name is also the voltage area name. For this case, do not specify a cell list in the `create_voltage_area` command. The cells that were specified as belonging to the power domain are the cells that the tool places in the given voltage area. Using the `-power_domain` option ensures correct alignment of the logic hierarchies with the voltage areas.

If the design does not contain power domains or you do not use the `-power_domain` option, you must then specify a voltage area name and provide the list of cells that are to be placed in the given voltage area. In this case, you are responsible for making sure that the hierarchies align correctly with their voltage areas.

For an example on how to use the `create_voltage_area` command without using the `-power_domain` option, see the script provided in [Example 12-1 on page 12-17](#). For an example that uses the `-power_domain` option, see the script provided in [Example 12-3 on page 12-33](#).

Note:

You can specify guard bands when creating voltage areas by using the `-guard_band_x` and `-guard_band_y` options with the `create_voltage_area` command. See “[Including Guard Bands With Voltage Areas](#)” on page 12-18.

The script in [Example 12-1](#) shows you how to create voltage areas manually without specifying power domains.

Example 12-1 Manual Voltage Area Specification

```
# Block1 is VoltageArea1 and Block2 is VoltageArea2 #
##
create_voltage_area -coordinate {...} -name VoltageArea1
update_voltage_area -voltage_area VoltageArea1 Block1
create_voltage_area -coordinate {...} -name VoltageArea2
update_voltage_area -voltage_area VoltageArea2 Block2
report_voltage_area -all
```

Using the `-polygons` option you can specify a list of polygons for the voltage area. The geometry of the voltage area can be either a rectangle or a rectilinear polygon. This option is used for automatic shaping of the voltage area.

Updating Voltage Areas in a Design

After you have created voltage areas and associated hierarchical cells with them, you can use the `update_voltage_area` command to add or remove cells from specified voltage areas to adjust the utilization of the voltage area. However, you might prefer to change the size of the voltage areas and keep the utilization the same.

You can also use the `update_voltage_area` command to

- Add regions to the voltage area

When you add new regions, they should not overlap the existing regions of the voltage area. Note that you cannot remove a region from the voltage area using this command.

- Specify guard bands for the voltage area

To specify guard bands when updating voltage areas, use the `-guard_band_x` and `-guard_band_y` options with the `update_voltage_area` command. For more details, see “[Including Guard Bands With Voltage Areas](#).”

Default Voltage Area

The IC Compiler tool automatically derives a default voltage area after you create the first voltage area. Conversely, when you delete the last voltage area of the design, the tool automatically removes the default voltage area. The default voltage area is used for the placement of cells that are not specifically assigned to any voltage area. The default voltage area can contain top-level leaf cells and buffer-type level shifters, as well as blocks.

You can prevent the insertion of new cells in the default voltage area by setting the `mv_no_cells_at_default_va` variable to `true`. The default for this variable is `false`. Setting the `mv_no_cells_at_default_va` variable to `true` eliminates the requirement for

setting the `dont_touch` attribute on the affected top-level nets in the default voltage area. It also controls the placement of new buffers in the default voltage area. This variable is effective for designs with both abutted and nonabutted voltage areas at the top level.

To create a collection of voltage areas in the design, including the default voltage area, use the `get_voltage_areas` command.

To update the guard band information for the voltage areas, including the default voltage area, use the `update_voltage_area` command.

To report all the voltage areas in the design, including the default voltage area, use the `report_voltage_area` command.

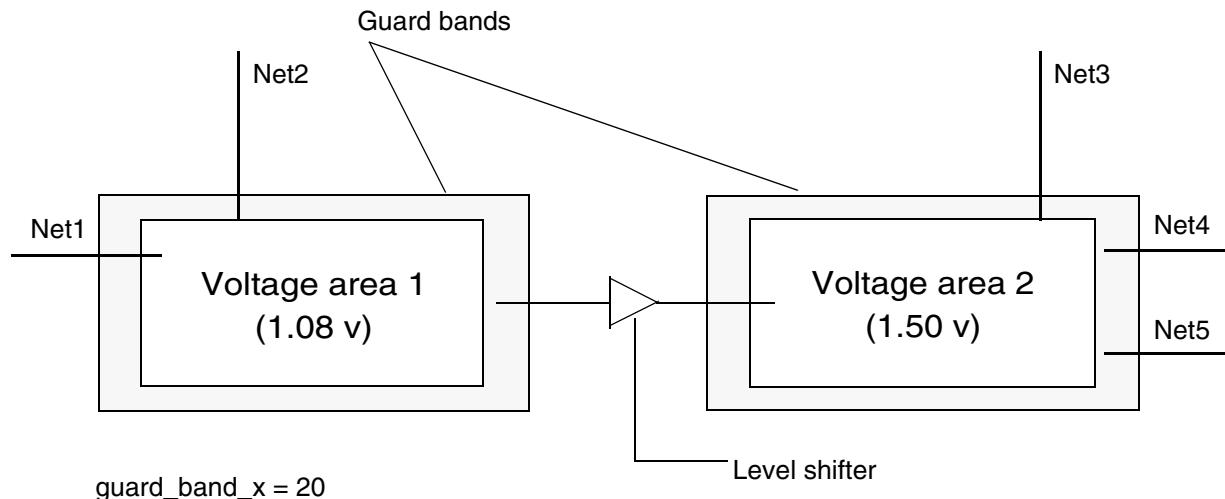
Including Guard Bands With Voltage Areas

You can use guard bands to ensure that no shorts occur at the boundaries of the voltage areas. Guard bands define hard keepout margins surrounding the voltage areas. No cells, including level shifters and isolation cells, can be placed within the guard band margins.

For example, using guard bands is recommended when the rows are the same for all the voltage areas. The guard bands guarantee that the cells in different voltage areas are separated so that power planning does not introduce shorts.

You can include guard bands when you create or update voltage areas. You generate guard bands by using the `-guard_band_x` and `-guard_band_y` options with the `create_voltage_area` or `update_voltage_area` command.

[Figure 12-5](#) shows two voltage areas with guard bands of different widths. For both voltage areas, the x-width is 20 and the y-width is 10. Notice that the x-width defines the horizontal width of the vertical part of the guard band, and the y-width defines the vertical width of the horizontal part.

Figure 12-5 Two Voltage Areas With Guard Bands

In the IC Compiler GUI, voltage areas are outlined. Spaces or margins between voltage areas indicate the presence of guard bands. The IC Compiler GUI can highlight the guard bands.

When generating guard bands by using the `create_voltage_area` or `update_voltage_area` commands, you should be aware of the following conditions:

- The guard band of a new voltage area can overlap an existing voltage area.
- The x guard band width applies to both the left and right side of a voltage area and the y guard band width applies to both the top and bottom side of a voltage area.
- Overlapping guard bands are not allowed.
- A guard band that lies outside the core area is allowed and does not generate a warning message.

Handling Nested Voltage Areas

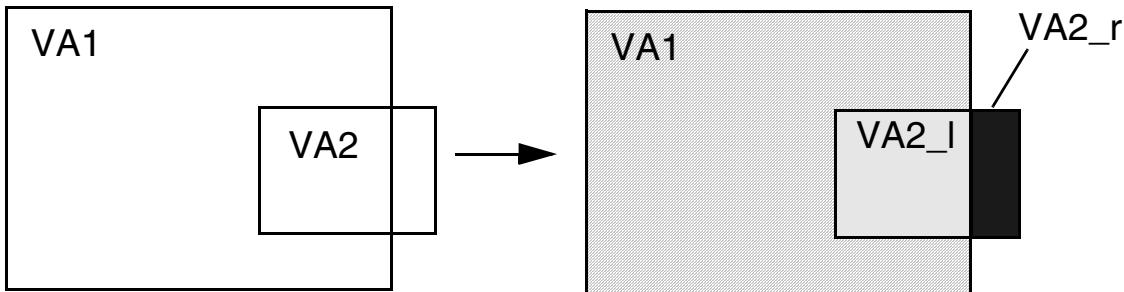
Voltage areas can be physically nested corresponding to the nested relationship of the logic hierarchy of nested power domains. You still use the `create_voltage_area` command to define the individual voltage areas. A given voltage area can contain one or more nested voltage areas. Note that a voltage area must completely contain its nested voltage areas. Intersecting voltage areas are not supported.

In the case where logically nested hierarchies would lead to overlapping voltage areas, you can use the `create_voltage_area` command to resolve the overlapping areas by defining a number of voltage areas consisting of separate, *abutted* rectangles or rectilinear shapes.

Each shape would share at least one boundary with another shape of the given voltage area.

[Figure 12-6](#) shows an example of areas intersecting and how you might resolve them into three physically abutted, separate voltage areas named VA1, VA2_l, and VA2_r. Voltage area named VA1 is an eight-sided rectilinear area, and voltage areas named VA2_l and VA2_r are rectangles.

Figure 12-6 Example of Physically Resolved Voltage Areas



Voltage Area Support for Macros

Macro cells can be listed as belonging to power domains and can be physically located within the corresponding voltage areas. If a macro is not listed in a power domain, it can be explicitly listed in the `create_voltage_area` command.

Note:

Care must be taken to define whether a given macro cell is internally isolated or requires external isolation. Internally isolated macros should have the `connected_to_iso_cell` attribute set on the appropriate library pin; otherwise the tool assumes external isolation might be required and adds an isolation cell to the appropriate input net. Usually, if attribute marking is required, it is done as part of logic synthesis.

Because a macro cell is a physical cell, that has a height, width, and location, a new voltage area must be large enough to contain the macro. Also, if a given macro cell already belongs to a voltage area, you cannot list that macro in any subsequent `create_voltage_area` command.

The `report_voltage_area` command reports all the top-level voltage area blocks, including macros. If a macro cell is within a hierarchical block that belongs to a voltage area, that macro is not part of the top-level blocks and is not reported.

Defining Always-On Power Wells Within Voltage Areas

To create an always-on well within a voltage area, use the `create_bounds` command with the `-exclusive` option to define a special placement area or well for specific cells within a voltage area. You define the location and dimensions of an exclusive move bound as well as providing the list of cells to be placed within the exclusive move bound. The geometry of the exclusive bound must fit within the voltage area. No cells except the specified cells can be placed within the bound.

After you define the exclusive move bounds, use the `set_power_guide` command to identify these bounds as power guides. The following example shows how to create an exclusive move bound and define it as an always-on power guide:

```
icc_shell> create_bounds -name AO_WELL -coordinate {10 10 20 20} \
    -exclusive [get_cells A_INST]
icc_shell> set_power_guide -name AO_WELL \
    -power_net VDD -ground_net GND
```

Use the `unset_power_guide` command to disable a move bound from being a power guide. You can determine the existing power guides by using the `report_power_guide` command.

You use the exception connection mechanism to connect these cells to the backup power net. The cells then function as always-on cells, so you can use single-power standard cells.

Removing Voltage Areas From the Design

Use the `remove_voltage_area` command to remove all voltage areas or specified voltage areas from the design. This includes removing any move bounds and guard bands that belong to the removed voltage area. If you remove a logically nested voltage area, the logic in the lower hierarchies inherits the voltage properties of the parent hierarchy.

Reporting on the Voltage Areas of a Design

To obtain information about existing voltage areas, use the `report_voltage_area` command. For each voltage area, the report includes the voltage area name, the list of hierarchical cells associated with the voltage area, and its geometry, area, utilization, and guard band information if guard bands are present.

Power Management Cells and UPF Strategies

This section discusses the requirements, properties, and commands you can use to manage the power management cells in your design.

- [Managing Level-Shifter Cells](#)
- [Managing Isolation and Enable-Type Level-Shifter Cells](#)
- [Retention Registers](#)
- [Always-On Logic](#)
- [Power Management Cells and the ASCII UPF Flow](#)

Managing Level-Shifter Cells

Level-shifter cells function as the interface between power domains that operate at different voltage levels. These cells ensure that the output transition of a driver can cause the receiving cell to switch even though the receiver is operating at a different voltage level.

The IC Compiler tool uses the strategies defined by the `set_level_shifter` command in the UPF file to insert the level-shifter cell. The details of inserting level shifters, specifying the threshold voltage, and specifying the level-shifter strategies are described in the following sections:

- [Inserting Buffer-Type Level Shifters](#)
- [Setting the Threshold Voltage for Level Shifters](#)
- [Defining the Level-Shifter Strategy](#)
- [Associating Specific Library Cells With a Level-Shifter Strategy](#)
- [Checking Level-Shifter Violations](#)
- [Placing Level Shifters](#)
- [Removing Level Shifters](#)

Inserting Buffer-Type Level Shifters

If buffer-type level shifters were not inserted during logic synthesis, you can use the `insert_mv_cells` or `insert_level_shifters` command to insert these level shifters in the current design.

Note:

If the design is synthesized using the Design Compiler tool, automatic level-shifter insertion is done as part of the `compile_ultra` command.

When you use the `insert_mv_cells` or `insert_level_shifters` command, level shifters are inserted on nets where significant voltage differences occur between drive pins and load pins. The voltage differences are determined either automatically by the tool or according to voltage threshold conditions that you define by using the `set_level_shifter_threshold` command. For more details, see “[Setting the Threshold Voltage for Level Shifters](#)” on [page 12-25](#).

Also, you can accept the default level-shifter strategy, which allows buffer-type level shifters that step up the voltage and those that step it down, or you can use the `set_level_shifter_strategy` command to control whether only step-up or step-down buffer-type level shifters are to be used. This command automatically assigns `dont_touch` attributes to the nets connecting ports and buffer-type level shifters.

Note:

Although you can manually insert buffer-type level shifters at a number of points in the multivoltage design flow, you should insert them early in the flow, preferably before logic synthesis, or let logic synthesis automatically insert them. The latest they should be inserted is just before you run the IC Compiler `place_opt` command.

By default, level shifters are not inserted on clock nets. To insert level shifters on clock nets use the `-all_clock_nets` or `-clock_net list_of_clock_nets` option with the `insert_level_shifters` command.

To obtain information about the level shifters available in the target libraries, use the `-verbose` option with the `insert_level_shifters` command. The information reported includes the library name and type, level-shifter cell name, operating conditions, input and output voltages, process ID, temperature, and tree type. The number of level shifters inserted is also reported.

The tool follows these steps to insert buffer-type level shifters in the voltage-mismatched nets:

1. Identifies the nets in the design with voltage mismatches that meet the automatically determined or user-defined threshold.
2. Analyzes the target library for available buffer-type level shifters, given the specified level-shifter rule.

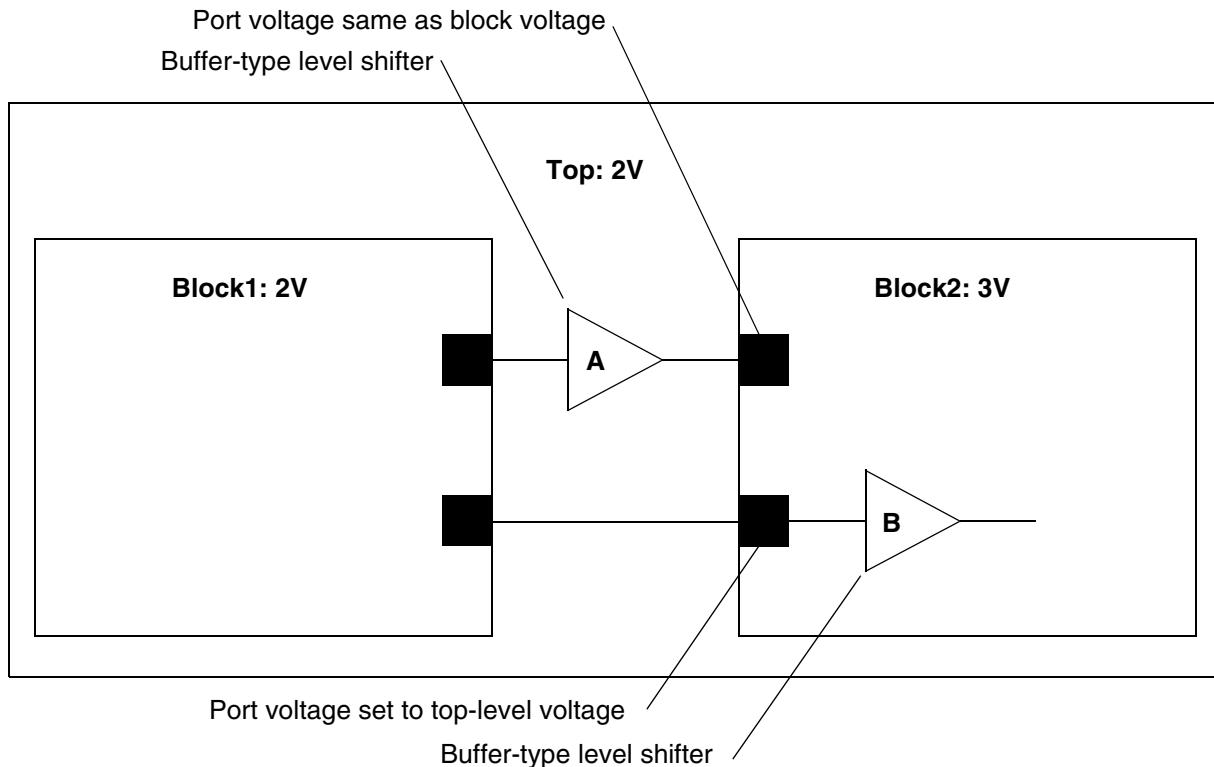
3. Instantiates the proper buffer-type level shifter, and sets a `size_only` attribute on the level shifters and a `dont_touch` attribute on the port-to-level-shifter nets.
4. Instantiates buffer-type level shifters in the clock nets as needed, but only if the `-all_clock_nets` or the `-clock_net` option is specified with the `insert_level_shifters` command.

The buffer-type level shifters are inserted between the blocks if the operating voltage is set on the instance and not on any of its ports. However, if an operating voltage different from the block operating voltage is set on a port of the block, the buffer-type level shifter for that port is inserted inside the block.

You annotate operating voltages on the instances or their hierarchical ports by using the `set_operating_conditions` command for blocks. To have a buffer-type level shifter inserted inside a block, use the `set_operating_conditions -object_list` command to provide a port list and operating voltage setting for these ports that is different from the block voltage.

[Figure 12-7](#) shows an example of buffer-type level-shifter insertion at both the top level for a block port that has the default block voltage and within a block for a port that is set at the top-level voltage.

Figure 12-7 Buffer-Type Level-Shifter Insertion at the Top Level and Block Level



Note:

Buffer-type level shifters are marked with the `size_only` attribute and therefore can be optimized as part of the physical implementation phase (the `place_opt` command).

Setting the Threshold Voltage for Level Shifters

A level-shifter threshold strategy must be established before buffer-type level shifters are inserted. The following are the two methods of establishing the threshold voltage on the level-shifter cells:

- The tool can automatically determine the default threshold values when inserting the buffer-type level-shifter cells.
- You can specify the threshold voltage for nets with voltage mismatches by using the `set_level_shifter_threshold` command.

Automatically Determined Threshold Voltage for Level Shifters

The tool can automatically determine level-shifter thresholds if the target library cells have the following attributes:

- V_{ih} – lowest input voltage for logic 1
- V_{il} – highest input voltage for logic 0
- V_{imax} – maximum input voltage for logic 1
- V_{imin} – minimum input voltage for logic 0
- V_{oh} – lowest output voltage for logic 1
- V_{ol} – highest output voltage for logic 0
- V_{omax} – maximum output voltage for logic 1
- V_{omin} – minimum output voltage for logic 0

These voltage parameters are used to determine input and output voltage bands (ranges) for both logic 1 and logic 0. For logic 1, the input voltage band is ($V_{imax} - V_{ih}$) and the output voltage band is ($V_{omax} - V_{oh}$). Similarly, for logic 0, the input voltage band is ($V_{il} - V_{imin}$) and the output voltage band is ($V_{ol} - V_{omin}$).

Therefore, for any pair of driver and load pins, if the output voltage band of the driver pin is completely overlapped by (contained within) the input voltage band of the load pin, no level shifter is required. This behavior holds for both logic 1 and logic 0.

Conversely, a sufficient voltage difference between the driver and load pins exists so that buffer-type level-shifter insertion is required if any of the following conditions hold:

- Driver pin $V_{o\max} >$ Load pin $V_{i\max}$
- Driver pin $V_{o\min} <$ Load pin $V_{i\min}$
- Driver pin $V_{oh} <$ Load pin V_{ih}
- Driver pin $V_{ol} >$ Load pin V_{il}

For any of these four conditions, the driver voltage band does not fall completely within the load voltage band, and buffer-type level shifters are needed.

Specifying the Threshold Voltage for Level Shifters

You can override the default threshold conditions by using the `set_level_shifter_threshold` command to specify the minimum voltage difference beyond which the voltage must be adjusted by using a buffer-type level shifter. Use the `-voltage` option to specify an absolute voltage difference between the source and sink voltages, or use the `-percent` option to specify the percentage difference.

You can specify both the options simultaneously. In this case, buffer-type level shifters are inserted when either threshold condition is met. For example, to specify a 0.5 voltage threshold and a 5 percent threshold, use the following command:

```
icc_shell> set_level_shifter_threshold -voltage 0.5 -percent 5
```

Note:

The percentage difference is computed as $100 \times (\text{absolute value of the difference between the driver voltage and the load voltage}) / (\text{driver voltage})$.

Defining the Level-Shifter Strategy

Use the `set_level_shifter` command to set the strategy for adjusting the voltage levels in the design. The IC Compiler tool follows the specified strategy to insert level shifters on signals that have sources and sinks that operate at different voltages. If you do not specify a level-shifter strategy for a particular power domain, the default level-shifter strategy applies to all elements in the power domain. For example, to define a level-shifter strategy which specifies that inserts level-shifters on all outputs of the power domain PD1 if the voltage difference exceeds 0.02, use the following command:

```
icc_shell> set_level_shifter ls_start1 -domain PD1 \
    -applies_to outputs -threshold 0.02
```

Associating Specific Library Cells With a Level-Shifter Strategy

By default, the tool maps the level-shifter cells to any suitable level-shifter cells in the logic library. To associate a specific set of library cells to the specified level-shifter strategy, use the `map_level_shifter_cell` command. When the tool inserts the level-shifter cells, it chooses the library cells that are specified with the `-lib_cells` option of the `map_level_shifter_cell` command. This command has no effect on instantiated level-shifter cells that have a `dont_touch` attribute set on them.

Checking Level-Shifter Violations

Use the `check_mv_design -level_shifters` or `analyze_mv_design -level_shifter` command to report net, cell, and environment violations and design details that can help in understanding the level shifters. You can run the command at any point in the flow. (These commands are preferred over the `check_level_shifters` command.)

Placing Level Shifters

Level shifters can be placed at regular sites (single-height cells) or at special sites (multiheight cells) with multiple power rails. For regular sites, you are responsible for connecting secondary power to the level shifters. For details, see “[Routing Secondary Power and Ground Pins](#)” on page 6-76.

If special sites are needed for level shifters, these sites must be instantiated in the floorplan at locations where these level shifters could be placed. Such sites are typically defined along the voltage area boundaries. They can overlap with the base site array.

Level-shifter placement is performed when you run the `place_opt` command or when you issue the `create_placement` and `legalize_placement` commands. If the library attribute `is_level_shifter` is set to `true` for the level shifters, the tool attempts to place the level-shifter cells at legal sites nearest the voltage area boundaries. This same placement process holds for isolation cells that have their `is_isolation_cell` attribute set to `true`.

Removing Level Shifters

Use the `remove_level_shifters` command to remove all level shifters in a design except those marked with the `dont_touch` attribute. Use this command to correct or redefine the use of level shifters in the design.

Managing Isolation and Enable-Type Level-Shifter Cells

In multivoltage designs, the IC Compiler tool inserts isolation cells when signals leave a powered-down block and enter a block that can be powered-up or is always-on. An isolation cell provides a known constant logic value to an always-on block when the power-down block has no power, thereby preventing unknown or intermediate values.

The tool uses the strategies defined by the `set_isolation` command in the UPF file to insert the isolation cells. The following sections provide details about inserting the isolation cells and specifying the isolation strategies:

- [Inserting Isolation Cells and Enable-Type Level Shifters](#)
- [Defining the Isolation Strategy](#)

Inserting Isolation Cells and Enable-Type Level Shifters

Isolation cells and enable-type level shifters are used to selectively shut off the input side of the voltage interface of a voltage area. Isolation cells do not shift the voltage. In general, enable-type level shifters are used to step the voltage up or down, but they can also be used as isolation cells.

Isolation cells can be instantiated at the RTL level or you can insert them by using the `insert_mv_cells` command, at various points in the flow. The enable-type level shifters can be instantiated only at the RTL level or inserted by the tool to replace isolation cells that are already present. You should instantiate isolation cells and enable-type level shifter cells at the RTL level; otherwise, formal verification of the logic is likely to fail.

If you insert your own power management cells, set the `is_isolation_cell` and `isolation_cell_enable_pin` attributes on the isolation cells and their enable pins. Similarly, set the `is_level_shifter` and the `level_shifter_enable_pin` attributes on the enable-type level shifter cells and their enable pins. After these cells are instantiated in the netlist, mark them with the `size_only` attribute before logic optimization.

Use the `check_isolation_cells` command to check for redundant isolation cells. For details about various multivoltage-specific checks, see “[Multivoltage-Specific Checks](#)” on page [12-43](#).

Defining the Isolation Strategy

Use the `set_isolation` command to define the isolation strategy for a power domain and its elements. For example, to define an isolation strategy named iso_strat1 that applies to both the input and output ports of a power domain named PD1, use the following command:

```
icc_shell> set_isolation iso_strat1 -domain PD1 -applies_to both \
           -isolation_power_net PN1 -isolation_ground_net GN1
```

Setting the Isolation Control

Every isolation strategy specified by using the `set_isolation` command must have a corresponding isolation control definition specified by using the `set_isolation_control` command, unless the strategy is `-no_isolation`.

The `set_isolation_control` command enables you to specify the isolation control signal and sense separate from the isolation strategy. The command identifies an existing isolation strategy and specifies the isolation control signal for that strategy. For example, to specify the isolation signal, isolation sense, and the hierarchical location of the isolation cell for the existing strategy named iso_strat1 of the power domain named PD1, use the following command:

```
icc_shell> set_isolation_control iso_strat1 -domain PD1 \
           -location parent -isolation_signal en -isolation_sense low
```

Retention Registers

In multivoltage designs, when a power domain is shut down and later powered up, the power domain must resume operation based on its last good state. Special cells called retention registers store the state during the shutdown.

Use the `set_retention` command to specify the registers in the power-down domain that should be implemented as retention registers. For example, to define a retention strategy for all the sequential cells in the power domain named PD1 use the following command:

```
icc_shell> set_retention retention_1 -domain PD1 \
           -retention_power_net PN1_B -retention_ground_net GN1
```

Every retention strategy specified by using the `set_retention` command must have a corresponding retention control definition specified by using the `set_retention_control` command. The `set_retention_control` command allows you to specify the retention control signal and sense separately from the retention strategy. The command identifies an existing retention strategy and specifies the save and restore signals and senses for that strategy. For example, to specify the save and restore signals and senses for the strategy named retention_1 of the power domain named PD1, use the following command:

```
icc_shell> set_retention_control retention_1 -domain PD1 \
           -save_signal {save_net high} -restore_signal {restore_net low}
```

The `map_retention_cell` command provides a mechanism for constraining the implementation choices for retention registers. This command requires the name of an existing retention strategy and power domain. For example, to specify that retention cells for the strategy named retention_1 of the power domain named PD1 should be implemented by using ret_cell1 or ret_cell2 library cells, use the following command:

```
icc_shell> map_retention_cell retention_1 \
           -domain PD1 -lib_cells {ret_cell1 ret_cell2}
```

Always-On Logic

Multivoltage designs typically have power domains that are shut down and powered up during the operation of the chip while other power domains remain powered up. The control nets that connect cells in an always-on power domain to cells within the shut-down power domain must remain on during shut down. These paths are referred to as always-on paths.

The target library can have special cells that can be used on the always-on paths and such cells are called the always-on cells. These library cells have the cell-level Boolean attribute `always_on` set to `true`, in the target library. These cells have the primary and backup power pins and are also referred to as dual-rail always-on cells. If your target library does not support the `always_on` attribute, you can set the `always_on` attribute on the single-rail standard cells in the library, using the `set_attribute` command. The IC Compiler tool uses these always-on cells during always-on optimization. It performs always-on synthesis only when the target library contains always-on buffers and always-on inverters.

For more information about handling always-on logic, see the *Power Compiler User Guide*.

Power Management Cells and the ASCII UPF Flow

In the ASCII UPF flow, when the design is synthesized using a third-party tool, the IC Compiler tool supports commands to associate the existing power management cells with the UPF strategies and if required, to insert new power management cells in the design. The following sections discuss these capabilities in more detail:

- [Inserting Power Management Cells](#)
- [Associating of Power Management Cells With UPF Strategies](#)

Inserting Power Management Cells

You can insert isolation, level-shifter, and retention cells in your design by using the `insert_mv_cells` command. This command uses the strategies defined in the UPF file to insert these cells. While inserting the new power management cells, the tool preserves the power management cells that already exist in the design. The `insert_mv_cells` command inserts the power management cells in the following order:

- Isolation cells
- Level-shifter cells
- Enable level-shifter cells

You can choose to insert only specific power management cells by using specific options of the `insert_mv_cells` command.

The following command options and command sequence results in the insertion of enable level-shifter cells:

- `insert_mv_cells -all`
- `insert_mv_cells -isolation -level_shifter`
- `insert_mv_cells -isolation
insert_mv_cells -level_shifter`
- `insert_mv_cells -level_shifter
insert_mv_cells -isolation
insert_mv_cells -level_shifter`

Associating of Power Management Cells With UPF Strategies

The IC Compiler tool supports the `associate_mv_cells` command to associate the existing power management cells in the design with the strategies defined in the UPF file. This command also checks and reports the correlation between the power management cells and the UPF strategies. The command reports the following information about the level-shifter cells and level-shifter strategies:

- The number of level-shifter cells that are associated with each level-shifter strategy, including the default strategy.
- The number and names of the level-shifter cells that do not have any supply net connection.
- The number and names of level-shifter cells that violate the level-shifter strategy.
- All level-shifter cells and their associated strategy information, if you use the `-verbose` option.

N-Well and P-Well Bias

Some process technologies allow dedicated voltage supplies, instead of normal rail voltages, to be applied to n-well and p-well regions of the chip. Applying a bias voltage to a well changes the threshold voltage for transistors in the well, which affects the performance and leakage current.

The IC Compiler tool offers an optional mode to specify the n-well and p-well bias supply infrastructure using UPF commands. In this mode, the tool automatically makes supply connections to the n-well and p-well bias pins.

To enable the UPF-based well bias mode, set the `enable_bias` design attribute to `true` in the UPF command file as follows:

```
set_design_attributes -elements {} -attribute enable_bias true
```

With the `enable_bias` design attribute set to `true`, supply sets and supply set handles can be used to specify the bias supply connections. The well bias pins, along with the power supply and ground pins, are connected automatically for standard cells, macros, and other types of cells.

For more information about using this mode, see “N-Well and P-Well Bias” in the *Synopsys Multivoltage Flow User Guide*.

Example IC Compiler Script for the UPF Flow

The following is an example script for power intent specification using the UPF commands.

Example 12-2 Example UPF Script for Power Intent Specification

```
# Use the load_upf command to load this file

set_scope
name_format -isolation_prefix "ISO_" -level_shift_prefix "LS_"
create_power_domain TOP
create_supply_port VDD -domain TOP
create_supply_port VDDI -domain TOP
create_supply_port VDDG -domain TOP
create_supply_port VDDM -domain TOP
create_supply_port VSS -domain TOP
add_port_state VDD -state {active_state 1.0}
add_port_state VDDI -state {high_state 1.0} \
    -state {low_state 0.8} -state {off_state off}
add_port_state VDDG -state {high_state 1.0} \
    -state {low_state 0.8} -state {off_state off}
add_port_state VDDM -state {active_state 1.0}
create_supply_net VDD -domain TOP
create_supply_net VDDI -domain TOP
create_supply_net VDDIS -domain TOP
create_supply_net VDDG -domain TOP
create_supply_net VDDM -domain TOP
create_supply_net VSS -domain TOP
set_domain_supply_net TOP -primary_power_net VDD -primary_ground_net VSS
create_power_switch my_sw -input_supply_port {vin VDDI} \
    -output_supply_port {vout VDDIS} -control_port {ctrl inst_on} \
    -on_state {state1 VDDG {inst_on}}
map_power_switch my_sw -domain TOP -lib_cells HDRDID2HVT
connect_supply_net VDDI -ports {VDDI inst_sw/vin }
connect_supply_net VDDIS -ports {inst_sw/vout InstDecode/VDDIS}
connect_supply_net VDD -ports {VDD GPRs/VDD MULT/VDD}
connect_supply_net VDDG -ports {VDDG GPRs/VDDG}
connect_supply_net VDDM -ports {VDDG Multiplier/VDDM}
connect_supply_net VSS \
    -ports {VSS InstDecode/VSS GPRs/VSS Multiplier/VSS}
create_pst chiptop_pst -supplies {VDD A/VDDMS B/VDDIS C/VDDGS}
add_pst_state s0 -pst myblock_pst -state {1.0 0.8 0.8 1.0}
```

```

add_pst_state s1 -pst myblock_pst -state {1.0 1.0 1.0 1.0}
add_pst_state s2 -pst myblock_pst -state {1.0 off 1.0 1.0}
add_pst_state s2 -pst myblock_pst -state {1.0 1.0 off off}

```

The following is an example of a IC Compiler script for the UPF flow.

Example 12-3 Example IC Compiler Script for the UPF Flow

```

# Invoke the IC Compiler tool

## Open design .ddc file
read_ddc design.ddc
save_mw_cel -as floorplan
save_upf out.upf
link_physical_library
## derive pg connections
derive_pg_connection -create_nets -reconnect

## Floorplan steps
read_pin_pad_physical_constraints design_ios.tcl

## Initialize floorplan
create_voltage_area -power_domain PD1 -guard_band_x 1 -guard_band_y 1
check_mv_design -verbose

## Power plan and secondary power pin routing steps
create_rectangular_rings -nets {VDDGS VDDMS}
create_rectangular_rings -nets {VSS }
create_power_straps -direction vertical -start_at x \
    -nets {VSS VDD VDDI} -layer M6 -width 2
create_power_straps -direction horizontal -{VSS VDD VDDI ...} \
    -layer M5 -width 2
preroute_standard_cells -mode net -nets VDDG \
    -h_layer M5 -v_layer M6 \
    -h_width 0.38 -v_width 0.38

## Switch cell mapping and insertion
map_power_switch -domain top -lib_cells HDRSODHVT name_of_switch
create_power_switch_array -lib_cell "GSWITCH" \
    -bounding_box {110 200 250 400}
## Connect switch cells in daisy/hfn mode
connect_power_switch -source inst_on -port_name on -mode hfn -verbose

## Implementation steps
place_opt
clock_opt
route_opt

## Chip finishing
set_cell_vt_type -vt_type "VtType1"
set_vt_filler_rule
insert_stdcell_filler -cell_without_metal
check_mv_design

```

```
## Write results out
save_upf out2.upf
## The Power Ground Netlist is written
write_verilog -pg -output_net_name_for_tie out.v

## This is the non power ground netlist
write_verilog file_name.v
```

Voltage-Area-Aware Capabilities

The `place_opt`, `clock_opt`, and `route_opt` commands are voltage-area-aware, which means that these commands can be used on multivoltage designs. Other important capabilities include

- [Automatic High-Fanout Synthesis](#)
- [Virtual Hierarchy Routing](#)
- [Voltage-Area-Based Optimization](#)
- [Voltage-Area-Based Clock Tree Synthesis and Optimization](#)
- [Voltage Area Feedthrough Flow](#)
- [Voltage-Area-Aware Always-On Synthesis](#)
- [Voltage-Area-Based Power and Ground Net Associations](#)
- [Hierarchical Flow](#)
- [Multivoltage Relative Placement Capability](#)

Automatic High-Fanout Synthesis

This capability is part of the `place_opt` command. Buffer trees are built with dedicated subtrees for the fanouts in each voltage area. In particular, automatic high-fanout synthesis selects buffers according to the associated operating condition. Buffers are inserted and placed correctly.

Virtual Hierarchy Routing

The virtual route estimator takes the presence of voltage areas into account. This estimator observes the following constraints:

- Virtual routes for nets connecting cells within a voltage area do not go outside the voltage area.
- Virtual routes for nets connecting cells outside a voltage area detour around the voltage area.
- Virtual routes for nets crossing a voltage area do not zigzag in and out of the voltage area: the number of boundary crossings is minimized.

Also, routing-driven synthesis, such as maximum net length optimization, uses voltage-area-aware routing.

Voltage-Area-Based Optimization

Cell placement and buffer optimization is one-pass voltage-area-aware. When the tool buffers a net that crosses voltage areas, the net is divided into multiple segments, each of which is confined within a single voltage area (including the default voltage area), and buffering is confined to the segments. The insertion of new buffers does not introduce additional zigzag paths that cross the voltage area boundaries.

For designs with multiple voltage areas, the tool ensures that a tie cell in one voltage area does not feed a cell in another voltage area. You should ensure that constant nets remain within their respective voltage area boundaries.

If the `direct_power_rail_tie` attribute has been set on a library pin, instances of this pin do not connect to constant signals by connecting to a tie-off cell. Instead, such pins are connected to generic constant signals so that during power routing these pins can be connected directly to power rails.

Voltage-Area-Based Clock Tree Synthesis and Optimization

Clock tree synthesis and optimization are voltage-area-aware; they honor the following constraints:

- Sink pins are separated and clustered by voltage areas so that clock subtrees are built for each voltage area.
- A guide buffer is inserted for the set of sink pins for each voltage area to ensure that any subsequent levels of clustering do not mix pins from different voltage areas.

After the clock subtrees are built for each voltage area, clock tree synthesis can proceed in the usual manner, joining the subtrees at the root of the clock net. In addition to the synthesis of the initial clock tree, the constraints that are listed previously are also honored by several clock tree optimization techniques, such as buffer relocation, buffer sizing, gate relocation, and delay insertion.

Voltage Area Feedthrough Flow

The router generates feedthrough routes that cross voltage areas. To create feedthrough nets for disjoint voltage areas, the following criteria must be met:

- The load and the drivers present in the two disjoint voltage areas must belong to different hierarchical modules.
- For a net within one disjoint voltage area, all the cells of the net must belong to the same hierarchical module.

Note:

Feedthrough ports are not generated for internal nets whose driver or driver-region cells and load or load-region cells belong to the same hierarchical module; however, the load and the driver are placed in two disjoint voltage areas.

Voltage-Area-Aware Always-On Synthesis

The IC Compiler tool optimizes always-on feedthrough nets that cross voltage areas due to differences in the logical and physical view of the design. The tool supports the insertion of both single-rail and dual-rail always-on buffers. However, the tool inserts regular buffers on always-on feedthrough nets, if doing so does not introduce electrical violations. Inserting regular buffers instead of always-on buffers improves QoR and reduces congestion.

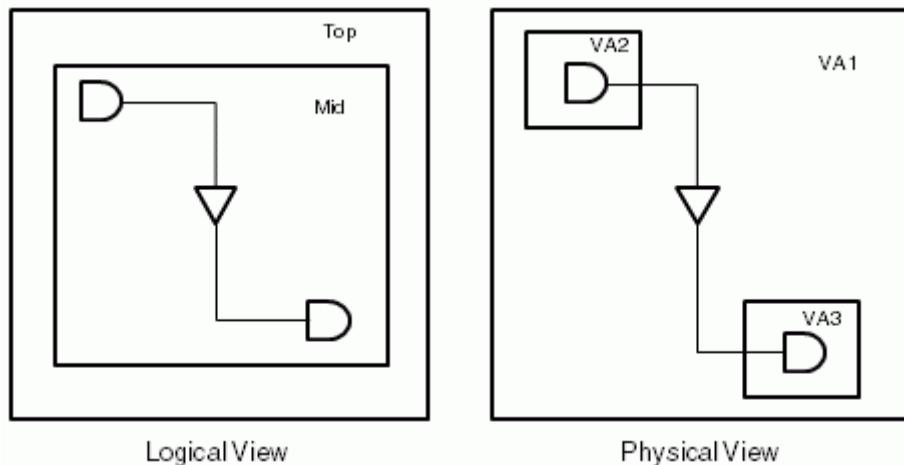
By default, the tool inserts regular buffers on always-on feedthrough nets, instead of always-on buffers, in the following situations:

- Based on the power state tables, the primary supply of the voltage area that the feedthrough net is crossing is equally or less always-on than the primary supply of the source and is equally or more always-on than the primary supply of the sink.
- The net is an isolation datapath without an isolation strategy at the source or sink.

You can force the tool to only use always-on buffers for always-on feedthrough nets by setting the `mv_make_primary_supply_available_for_always_on` variable to `false`, changing it from its default of `true`.

[Figure 12-8](#) shows an example of an always-on buffer added to an always-on feedthrough net. In the logical view, the net and the buffer are in the same hierarchy Mid, which is an always-on domain. However, in the physical view, the voltage area associated with this power domain is in two disjoint voltage areas.

Figure 12-8 Always-On Buffer Insertion in Disjoint Voltage Areas

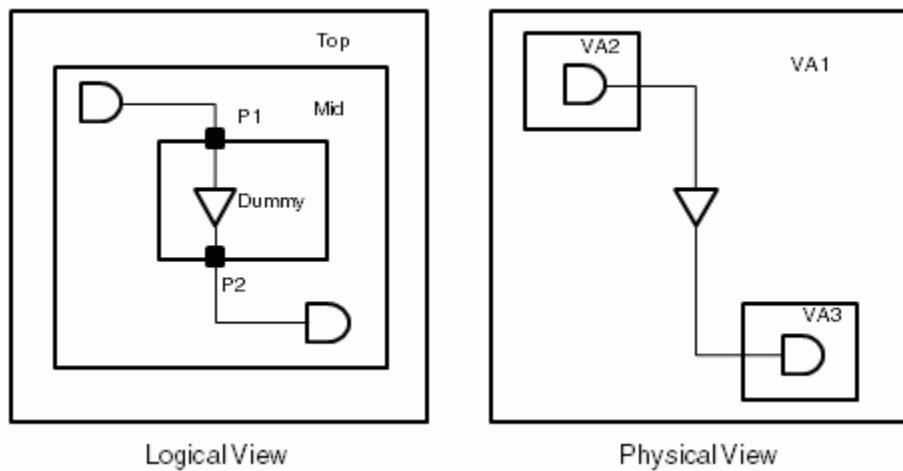


If the primary supply defined in the power domain for the subdesign Mid is available in the power domain for Top, the tool inserts an always-on buffer in the subdesign Mid that physically belongs to the Top design.

The tool follows these steps to support always-on synthesis across disjoint voltage areas:

1. Creates a dummy logical hierarchy inside the existing hierarchy Mid as shown in [Figure 12-9](#).
2. Creates two hierarchical ports, P1 and P2, on the dummy hierarchy and connects the buffer inside the dummy hierarchy to these ports.
3. Associates the dummy hierarchy with the existing voltage area, VA1, to which the buffer belongs.

Figure 12-9 Creating Dummy Hierarchy to Support Always-On Buffer Insertion in Disjoint Voltage Areas



Controlling the Dummy Hierarchy

The creation of the dummy logical hierarchy and port punching on the dummy hierarchy allow the tool to perform always-on synthesis and legalization. You can associate the dummy hierarchy with the default voltage area if the buffer belongs to the default voltage area.

Controlling the Location

You can control the location of the dummy hierarchy by setting the `skip_vao_hierarchy` attribute. Dummy hierarchies are not created in the hierarchical cells that have the `skip_vao_hierarchy` attribute. For example, to skip the creation of dummy hierarchy in the hierarchical cell A, you set the `skip_vao_hierarchy` attribute as shown in the following example:

```
icc_shell> set_attribute [get_cells A] skip_vao_hierarchy true
```

The `skip_vao_hierarchy` attribute is persistent in the IC Compiler session where it is created and the attribute is saved in the Milkyway database.

Controlling the Name

You can specify a prefix to distinguish the tool-created dummy hierarchies from the other hierarchies in the design. The new dummy hierarchies created by the `psynopt` and `place_opt` commands to fix DRC violations start with the specified prefix.

Use the `vao_feedthrough_module_name_prefix` variable to specify the prefix. The following example sets the `vao_feedthrough_module_name_prefix` variable to `SNPS_`. With this variable setting, the names of all the newly created dummy hierarchies start with `SNPS_`.

```
icc_shell> set_app_var vao_feedthrough_module_name_prefix SNPS_
```

Voltage-Area-Based Power and Ground Net Associations

You can assign specific power and ground tie-off nets to voltage areas by using the `set_power_net_to_voltage_area` command. This command allows you to link and unlink these nets to the specified voltage areas. Using the `-report` option, you can determine the power and ground nets currently associated with a voltage area. In addition, the CEL consistency checker automatically checks the multivoltage (voltage area) tie-off cells against the assigned settings.

Hierarchical Flow

Block abstractions are supported in multivoltage design optimization flows. These hierarchical models are used to model the interface logic of the mapped lower-level blocks of a hierarchical design. In general, the noninterface logic is excluded from the model, although you can specify options that allow certain noninterface nets and cells to be included. For more information about hierarchical models, see [Chapter 10, “Using Hierarchical Models.”](#)

The following relationships between voltage areas and hierarchical models are supported:

- A voltage area of the top-level design can contain hierarchical models. The combined physical area of the hierarchical models must be less than or equal to the area of the voltage area.
- A hierarchical model at the top level can contain voltage areas.

In both cases, the operating conditions applied to the block by using the `set_operating_conditions` command before you create the hierarchical model for the block are not visible at the top level. At the top level, use the `propagate_constraints` command with the `-operating_conditions` option to propagate the operating conditions placed on the design, cells, and ports of the model, and on the hierarchical pins inside the model.

Note that propagation of voltage areas inside a model is not supported. Therefore, you must propagate the operating conditions of the hierarchical cells within the voltage area by using the `propagate_constraints` command, or you must reapply the operating conditions from the top level.

Level shifters are kept only if they are an actual part of the interface logic. Those level shifters past the first registers of a block are not retained. For level shifters that become part of the hierarchical model, the leaf cells connected to the input and output nets of the level shifter are also retained, as well as the nets. You can use the `check_level_shifters` command to determine whether any output nets of the level shifters are floating.

Multivoltage Relative Placement Capability

You can use the relative placement capability with multivoltage designs. Relative placement is most often applied to physical datapaths. It provides a way for you to create structures in which you specify the relative column and row positions of instances with respect to each other. During placement and legalization, these structures are preserved, and the cells in each structure are placed as a single entity.

First, you group selected cells of a voltage area into a relative placement grouping. You can then place them as a single structure within the voltage area. The flow is identical to a typical relative placement flow, and the entire relative placement feature set is available.

A relative placement group should not span voltage areas. If it is necessary to span voltage areas, break the relative placement group into multiple relative placement groups, each contained within its voltage area.

In situations where a relative placement group is placed very close to a voltage area boundary, legalization might place part of the relative placement group outside the multivoltage area. When this happens, the relative placement group is broken in a way that respects the voltage area constraint.

Placing and Optimizing the Design

When placing and optimizing the locations of the cells of the design, the `place_opt` command honors the power supply intent specified in the UPF file. The command places level shifters and isolation cells near the boundaries of defined voltage areas. No buffering is done between the placed level shifter and the edge of the voltage area.

To perform always-on synthesis, the tool first identifies the always-on paths connecting the drive pins in the always-on power domains to the always-on load pins in the shut-down power domains. Always-on paths can be determined only if the tool finds always-on pins in the shut-down power domains. The enable pins of the special dual-power cells and the input

pins of the single-powered, standard cells placed in always-on power wells are recognized as always-on pins. You can manually mark other pins as always-on if needed.

After the tool has established the always-on paths, it carries out optimizations appropriate to these paths, such as sizing and buffering. The tool uses ordinary cells for the portion of the paths outside the shut-down power domain. Inside the shut-down power domains, the tool optimizes paths using the cells that you have marked as always-on.

The tool uses the following rules during optimization:

- Always-on logic is mapped only with always-on cells.
- Always-on cells are used in the shut-down power domains and on the always-on nets derived by the tool during supply-net-aware always-on synthesis.
- Any gate with input pass gate pins that is driven by an always-on cell is swapped with a functionally equivalent cell that does not have pass gate pins at the inputs.

Note:

After completing always-on synthesis, you can determine the always-on logic by using the `get_always_on_logic` command. This command shows you the nets and cells on the always-on paths of the design.

Clock Tree Synthesis in Power-Down Domains

When performing clock tree synthesis, the `compile_clock_tree`, `clock_opt`, and `optimize_clock_tree` commands honor the isolation cells in the clock path and perform synthesis on the always-on clock paths.

To ensure correct isolation between shutdown and power-up power domains, clock tree synthesis does not insert buffers between the isolation cells and the power domain boundary.

The tool uses the following approach on the always-on clock paths during clock tree synthesis:

- Dual-power always-on cells are inserted in the shutdown power domain.
- The buffers inserted on the always-on paths are dual-power always-on cells, and these cells are in the powered-down power domain.
- The always-on cells are inserted in the correct power domain, that is, the always-on region.
- Redundant dual-power always-on cells are not inserted in the powered-down region or power domain.

Alternatively, you create always-on islands in the shutdown power domain and the tool inserts buffers in these always-on islands during clock tree synthesis. This ensures that the always-on paths are not disturbed during clock tree synthesis.

Note:

To avoid long runtimes, before performing clock tree synthesis, you should manually insert buffers at the top level if your clock path at the top level traverses from the output of one module to the input of another module.

You should use the `check_mv_design` command before and after the clock tree synthesis process.

The following example shows the usage of the `check_mv_design` command before and after the clock tree synthesis process:

```
open_mw_lib library
open_mw_cel cel
remove_clock -all
create_clock -name clk -period 10 -waveform {0.0 2.0 4.0 6.0}
set_clock_tree_options -clock_trees [get_ports clk] \
    -max_transition 0.5 -target_skew 0.1
check_mv_design
compile_clock_tree -clock_trees clock_name
check_mv_design
save_mw_cel -as post_cts
```

Multivoltage-Specific Queries, Checks, and Reports

The IC Compiler tool supports commands to query, check, and report multivoltage specific details. These commands enable you to

- Query and report multivoltage-specific `dont_touch` attributes.
- Find or query objects such as nets and ports using the UPF query commands such as `find_objects`, `query_cell_instances`, `query_cell_mapped`, `query_net_ports`, `query_port_net`.
- Check and report target library subsetting inconsistencies.
- Check and report the power management cells, strategies, and their consistency.
- Report the operating conditions and process, voltage, and temperature (PVT) values on different blocks.
- Report the power supply nets and ports and their connections across various blocks.
- Report the power switches and the power states in the design.

Querying and Reporting Multivoltage-Specific `dont_touch` Attributes

The IC Compiler tool supports the following commands to query and report multivoltage-specific `dont_touch` attributes:

- `get_dont_touch_nets`

You can create a collection of nets with `dont_touch` attributes that meet specific multivoltage-specific reasons or criteria by using the `-type mv` option with the `get_dont_touch_nets` command.

- `report_dont_touch_net`

You can report the nets that have `dont_touch` attributes because of multivoltage reasons by using the `-type mv` option with the `report_dont_touch_net` command.

- `report_net`

You can use the `report_net` command to identify the nets that have a `dont_touch` attribute because of multivoltage reasons based on the `mvd` that is printed in the attributes column of the report.

Multivoltage-Specific Checks

The following commands perform multivoltage-specific checks. You can use them at various stages of your multivoltage design implementation.

- `check_mv_design`

Use this command to check for various types of violations such as inconsistent and conflicting library settings, missing isolation cells, incorrect voltage shifting across power domains. Use the `-verbose` option to get details of the violations and inconsistencies.

- `analyze_mv_design`

Use this command to analyze and report multivoltage design issues. This command reports the design details that can help you understand the level-shifter and always-on violations. The command also reports design setup details, library information, and details of UPF constraints.

- `check_level_shifters`

Use this command to check if the level shifters in the design violate the specified level-shifter strategy. Use the `-verbose` option to get a detailed report of the violations.

- `check_isolation_cells`

Use this command to get a list of isolation cells in the design. However this command also reports the possible requirement of an isolation cell as well as redundant isolation cells, if any, in the design.

Multivoltage-Specific Reports

This section describes a list of multivoltage-specific reporting commands that provide details of the operating conditions, inconsistencies if any, statistics of the various blocks in your design, and so on.

- `report_power_domain`

You use this command to get specific details such as the scope and the operating condition of a power domain.

- `report_voltage_area`

You use this command to get a list of voltage areas in your design. You can also get the operating condition and the pin connection details of the voltage areas. If your design contains disjoint voltage areas, this command also reports the components of a disjoint voltage area.

- `report_level_shifter`

You use this command to report all the level-shifter cells in the design. Use the `-verbose` option to get details of the level-shifter strategy of the level-shifter cells.

- `report_isolation_cell`

You use this command to get details of the isolation cells in your design. Use the `-verbose` option to get details of the isolation strategy.

- `report_retention_cell`

You use this command to get the details of the retention cells in the design. Use the `-verbose` option to get details of the retention strategy.

- `report_supply_port`

This command reports the details of all or the specified power supply ports in the design.

- `report_supply_net`

This command reports the details of power supply nets in the design. If you use the `-include_exception` option, you get the details of the exception pins on the power supply net.

- `report_power_pin_info`

This command reports the power pin information for the instantiated cells and not the library cells, in the design. This command can also be used to get details of exception supply connections.

- `report_power_switch`

This command reports all the power switches in the design. Use the `-verbose` option to get details of the power switches.

- `report_pst`

This command reports the power states in the current design.

The following commands can also be useful in reporting multivoltage specific details.

- `report_timing`

You can use the `-voltage` option to report the operating voltage specified in the operating condition, for each path element.

- `report_operating_conditions`

This command reports all or the specified operating conditions in the specified target library.

- `report_target_library_subset`

This command reports the target library subsets set on the design.

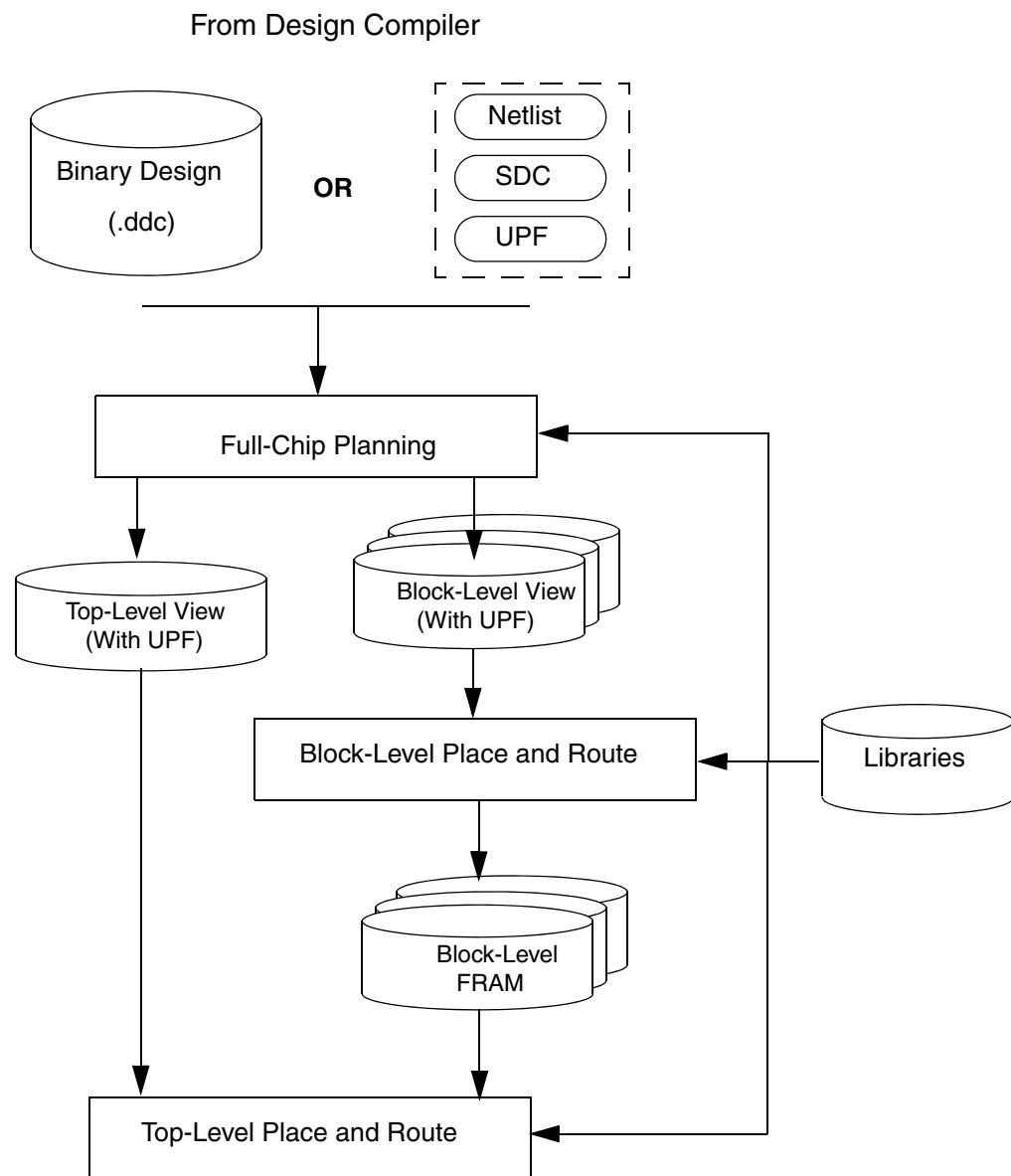
- `report_pg_net`

This command reports the power and ground net information for the opened Milkyway design.

Hierarchical UPF Flow

The IC Compiler tool supports the complete hierarchical methodology for the UPF flow. You can use the synthesized netlist obtained from the Design Compiler hierarchical UPF methodology and continue the hierarchical flow using the IC Compiler tool. [Figure 12-10](#) shows the IC Compiler hierarchical UPF flow.

Figure 12-10 IC Compiler Hierarchical UPF Flow



The following sections describe the hierarchical UPF flow in the IC Compiler tool:

- [Design Compiler to IC Compiler Interface](#)
- [Guidelines for the Hierarchical UPF Flow](#)
- [Full-Chip Design Planning](#)
- [Block-Level Place and Route](#)
- [Top-Level Place and Route](#)

Each of these steps is described in detail in the sections that follow.

Design Compiler to IC Compiler Interface

Both the Design Compiler and IC Compiler tools support the hierarchical UPF flow. After synthesizing your design using the Design Compiler hierarchical UPF flow, you can choose from the following two interface formats to continue the flow in the IC Compiler tool:

- A binary design database in .ddc format
- An ASCII design database in the Verilog format along with constraint files such as SDC and UPF.

The binary design database is recommended because it contains all information, including the constraints, that are to be transferred from the Design Compiler tool to the IC Compiler tool.

Guidelines for the Hierarchical UPF Flow

To use the IC Compiler hierarchical flow, follow the guidelines for methodology, interface, and implementation described in the following sections:

- [Methodology Guidelines](#)
- [Interface Guidelines](#)
- [Implementation Guidelines](#)
- [Limitations](#)

Methodology Guidelines

Follow these guidelines, which are common to both the Design Compiler and IC Compiler tools, to implement your design in the IC Compiler tool.

- Align the physical partitions with the power domain boundaries as defined in the logical netlist.
- Create all the necessary power supply nets for the power domain inside the power domain. These supply nets should also be connected from the top level.
- Specify the voltage information for each supply net.
- Specify the timing constraints.

Interface Guidelines

The first step to hierarchical design is determining the physical partitioning. Follow these interface guidelines while partitioning your design:

- The scopes of all the power domains inside a partition must be contained inside the partition.
- For all supply connections inside a partition, the supply nets must be specified within the partition.

Implementation Guidelines

You should follow these guidelines to implement your design using the hierarchical UPF flow:

- UPF characterization

To categorize the UPF information in the subblocks, you must run the `allocate_fp_budgets` command before running the `commit_fp_plan_groups` command.

When you run the `allocate_fp_budgets` command, it sets the related supply net on the ports of the partition, based on the following criteria:

- The direction of the port
- The location constraint of the isolation and level-shifter strategies
- The type of the pin: load or driver

While setting the related supply net, the tool performs checks for voltage violations, availability of supply nets, unspecified level-shifter strategies, and conflicts between isolation and level-shifter strategies.

When the subblock contains supply ports without PG pins, the `allocate_fp_budgets` and the `commit_fp_plan_groups` commands add dummy supply ports.

- Automatic propagation of the UPF from the hierarchical model (block abstraction) to the top level.

All the UPF constraints from the top level are applied only to the block. These constraints cannot be applied to the hierarchical model. However, the constraints are automatically propagated from the hierarchical model. So, do not reapply the UPF constraints or re-create the voltage area for the hierarchical model.

The `allocate_fp_budgets` command supports the following flexible block partitioning useful for the hierarchical flow.

- Partitioning of blocks that have power domains defined with `parent` as the scope. If you have two or more power domains defined on the same block,
 - The primary supply nets of the power domains must be electrically identical.
 - You can define a switch at the interface of the power domains only if a similar switch is present in the power domain of the top-level design.

Power Domain Merging

When you merge a power domain to the top level, the supply sets, nets, and ports of the power domain must be equivalent to those of the top-level power domain. In addition, the connectivity of the power domain should be equivalent to the top level.

During integration, the block-level ports that have the `snps_derived` UPF attribute are substituted by their equivalent top-level ports and supply nets or supply sets. The block-level supply net or supply sets are deleted. If the port is not marked with the `snps_derived` UPF attribute, the ports are not deleted.

If a power switch exists in the blocks, a matching switch must exist at the top level for the domains to be merged. It is an error to match a switch from the block to a switch in the top level that is already matched.

The following rules apply when merging domains that contain switch cells:

- A switch in the top level must have a unique switch in the domain being merged. The switches being merged should also have equivalent
 - Input supply nets
 - Control and acknowledgement signals separated by buffer or inverter pairs

- Voltage settings on the output supply nets
 - Port states, including the state names, state value, primary domain, and so on
- Also, the output supply nets must have similar connectivity with the other supply nets in the design.
- If the block-level domain has multiple equivalent switches, the first matching switch is used.

Limitations

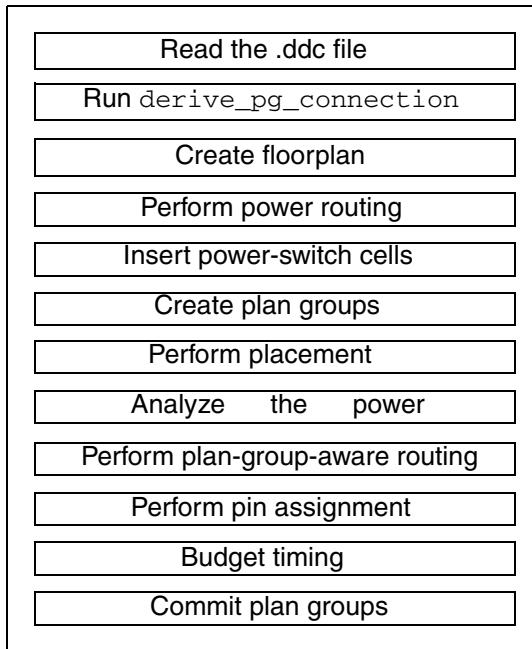
The following limitations apply to the IC Compiler hierarchical UPF flow:

- The target libraries that you use must contain enable level-shifter cells. The IC Compiler tool infers enable level-shifter cells when your design contains level-shifter cells that drive the isolation cells.
- You must define level-shifter strategies for all the power domains. The level-shifter strategy definition must specify the location as either `self` or `parent` on the inputs and outputs.

Full-Chip Design Planning

[Figure 12-11](#) shows the steps involved in the full-chip design planning phase. Use the `check_mv_design` command after every step to check for UPF consistency. You must follow the guidelines to successfully implement your hierarchical design. For more details about the guidelines, see [“Guidelines for the Hierarchical UPF Flow” on page 12-47](#) and the *IC Compiler Design Planning User Guide*.

Figure 12-11 Steps Involved in Full-Chip Design Planning



The following are the tasks involved in the full-chip design planning phase:

1. Read the .ddc file.

Read the .ddc files by using the `import_designs` command, as shown in the following example:

```
icc_shell> import_designs full_chip.ddc -format ddc \
           -top myDesign -cell myDesign
```

2. Create the power and ground net connections.

Using the `derive_pg_connection` command, create the power and ground net connections based on the power domain definitions and the power supply nets. This command performs automatic connection of power and ground pins.

```
icc_shell> derive_pg_connection -create_nets
icc_shell> derive_pg_connection -verbose
```

Use the `check_mv_design` command to check the power and ground connections.

3. Create the floorplan.

Using the `create_voltage_area` command, create the voltage areas for the power domains defined in the UPF file. Voltage areas are physical partitions corresponding to the power domains that are logical partitions that are defined in the UPF file.

```
icc_shell> create_voltage_area -power_domain myVA -guard_band_x 1 \
           -guard_band_y 1 -coordinate {60.140 60.140 275 244}
```

4. Perform power routing.

Create the power rings or straps manually before inserting the power-switch cells.

5. Insert the power-switch cells.

Before creating the switch cell, map the library power-switch cells to the corresponding switch cells in the UPF file by using the `map_power_switch` command. Then create the switch cells by using the `create_power_switch_array` command.

Connect the control signals of the switch cells by using the `connect_power_switch` command.

6. Create the plan groups.

Create the plan groups by using the `create_plan_groups` command. The partition boundary must correspond to the power domain boundary, as shown in the following example:

```
icc_shell> create_plan_groups -cycle_color \
           -coordinate "60.14 60.8 275 244.4" MyPlanGroup
```

7. Perform placement and power network analysis.

Run the initial placement and power network analysis for the power and ground nets. Set the resistance and pin layers before you run the `analyze_fp_rail` command.

8. Perform plan-group-aware routing and pin assignment.

Perform plan-group-aware global routing as shown in the following example. Do not allow feedthrough generation during pin assignment.

```
icc_shell> mark_clock_tree -clock_net
icc_shell> set_fp_pin_constraints -allow_feedthroughs off
icc_shell> set_fp_flow_strategy -plan_group_aware_routing true
icc_shell> set_route_zrt_common_options -plan_group_aware all_routing
icc_shell> route_zrt_global -exploration true
icc_shell> place_fp_pins -use_existing_routing[get_plan_groups *]
```

9. Perform proportional timing budgeting.

Use the `allocate_fp_budgets` command to generate the SDC file for the soft macro as shown in the following example. This command also characterizes the full-chip UPF constraints for the plan groups.

```
icc_shell> allocate_fp_budgets -file_format_spec {./sdc/m.sdc}
```

10. Transform the plan groups into soft macros.

Use the `commit_fp_plan_groups` command to transform the plan groups into soft macros.

```
icc_shell> commit_fp_plan_groups -push_down_power_and_ground_straps
icc_shell> save_mw_cel -hierarchy
```

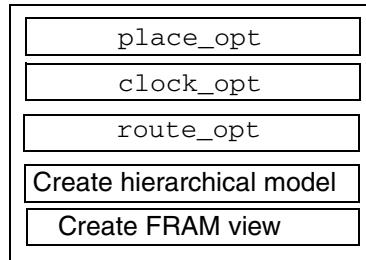
This completes the full-chip design planning phase. You then perform block-level place and route. See the following section, “[Block-Level Place and Route](#).”

Block-Level Place and Route

After you complete the full-chip design planning phase, the next phase in the hierarchical flow is the block-level place and route phase. For more details about the full-chip design planning phase, see “[Full-Chip Design Planning](#)” on page 12-50.

The block-level place and route phase involves the following tasks, as shown in [Figure 12-12](#):

Figure 12-12 Steps in the Block-Level Place and Route Phase



Before you perform the place and route step, check the UPF constraints inside the hierarchical model (block abstraction), as shown in the following example:

```
icc_shell> save_upf block.upf
icc_shell> report_power_domain
icc_shell> report_pst
icc_shell> report_supply_net
icc_shell> report_supply_port
```

Use the `check_mv_design` command to check the UPF consistency.

You then create the hierarchical model by using the `create_block_abstraction` command and create the FRAM view by using the `create_macro_fram` command. The following example creates the block abstraction and FRAM view for a block:

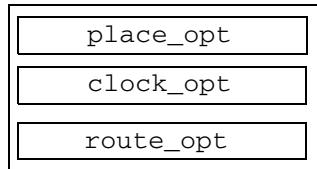
```
icc_shell> create_block_abstraction
icc_shell> create_macro_fram
```

Top-Level Place and Route

[Figure 12-13](#) shows the steps involved in the top-level place and route phase. In this phase, the UPF constraints in the block abstraction are propagated to the top level. When the blocks are integrated, the IC Compiler tool automatically propagates the UPF constraints from the block abstraction to the top-level when linking the design. So, do not use the `propagate_constraints -power_supply_data` command in the IC Compiler tool.

Use the `check_mv_design` command to check for UPF consistency.

Figure 12-13 Steps in the Top-Level Place and Route Phase



13

ECO Flow

An engineering change order (ECO) is an incremental change made to a complete or nearly complete design. You can use ECOs to fix functional, timing, noise, and crosstalk violations without synthesizing, placing and routing the entire design. You can also use ECOs to implement late-arriving design changes while maintaining design performance.

The IC Compiler ECO flows support both flat and hierarchical design methodologies. For hierarchical designs, you can change internal logic and ports on subblocks. However, you cannot delete or create hierarchical blocks or block abstractions. You must maintain the relationship between the logical and physical hierarchy.

The following topics describe the various ECO flows supported in the IC Compiler tool:

- [Unconstrained ECO Flow](#)
- [Freeze Silicon ECO Flow](#)
- [Hierarchical Placement Area Reservation ECO Flow](#)

The following topics describe the tasks you perform in the supported ECO flows:

- [Adding Spare Cells](#)
- [Making ECO Changes](#)
- [Placing Cells During the Unconstrained ECO Flow](#)
- [Placing and Mapping Cells During the Freeze Silicon ECO Flow](#)
- [Adding Repeater Cells on Routed Nets](#)

- [Performing ECO Routing](#)
- [Reporting ECO History and Physical Changes](#)
- [Comparing Designs](#)

Unconstrained ECO Flow

Use this flow if you have the flexibility to add new cells and move or delete existing cells. This flow is recommended if you have not taped out your design.

The unconstrained ECO flow consists of the following steps:

1. Update the design with the ECO changes by using the `eco_netlist` command, as described in “[Making ECO Changes](#).”
2. Update the power and ground connections by using the `derive_pg_connection` command.
3. For multivoltage designs, run the `check_mv_design` command and fix any violations.
If the ECO changes are not compliant with the UPF constraints, edit the UPF constraints to match the ECO intent.
4. Update the placement by using the `place_eco_cells` command, as described in “[Placing Cells During the Unconstrained ECO Flow](#).”
5. Update the routing by using the `route_zrt_eco` command or by manually routing the nets, as described in “[Performing ECO Routing](#).”

Freeze Silicon ECO Flow

Use this flow if your cell placement is fixed, and you can only change the metal and via mask patterns. This flow is recommended if you have taped out your design. To use this flow, your design must contain spare cells. You can add spare cells as described in “[Adding Spare Cells](#).”

The freeze silicon ECO flow consists of the following steps:

1. Update the design with the ECO changes by using the `eco_netlist` command, as described in “[Making ECO Changes](#).”
2. Update the power and ground connections by using the `derive_pg_connection` command.
3. For multivoltage designs, run the `check_mv_design` command and fix any violations.
If the ECO changes are not compliant with the UPF constraints, edit the UPF constraints to match the ECO intent.

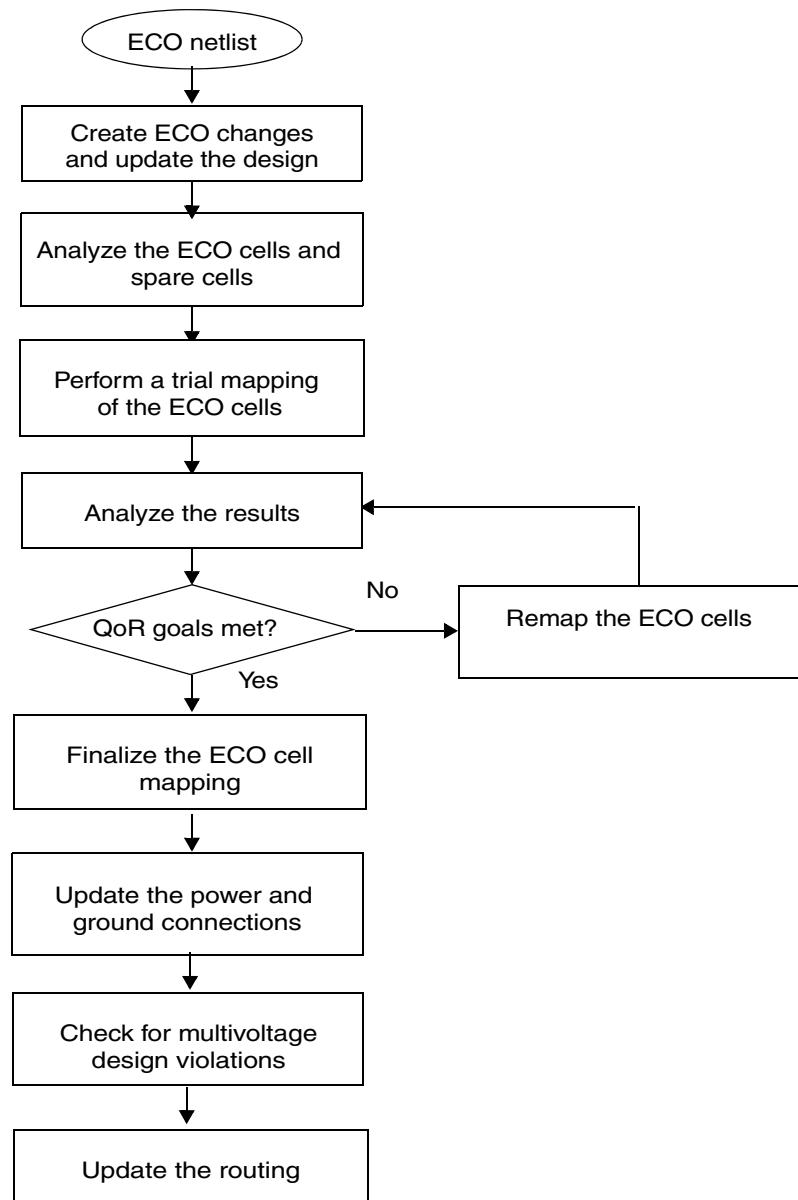
4. Map all the ECO changes to spare cells by using the `place_freeze_silicon` command. To manually map an ECO cell to a specific spare cell, use the `map_freeze_silicon` command, as described in “[Placing and Mapping Cells During the Freeze Silicon ECO Flow](#).”
5. Update the routing by using the `route_zrt_eco` command or by manually routing the nets, as described in “[Performing ECO Routing](#).”

Freeze Silicon QoR Tuning Flow

In the freeze silicon ECO flow, the tool maps each ECO cell to the nearest spare cell that has the same functionality. It does not consider QoR goals, such as timing, power, design rule violations, and congestion. To consider the design QoR when performing ECO mapping, use the freeze silicon QoR tuning flow. To use this flow, your design must contain spare cells. You can add spare cells as described in “[Adding Spare Cells](#).”

[Figure 13-1](#) gives an overview of the freeze silicon QoR tuning flow.

Figure 13-1 Freeze Silicon QoR Tuning Flow



Use the following steps to perform the freeze silicon QoR tuning flow:

1. Update the design with the ECO changes by using the `eco_netlist` command, as described in “[Making ECO Changes](#).”
2. Analyze the ECO cells and spare cells by using the `check_freeze_silicon` command.

You can visualize the distribution of the ECO cells and spare cells in the GUI. To create a collection of all the spare cells, use the `all_spare_cells` command. To create a collection of all the ECO cells, use the following command:

```
icc_shell> get_cells -hierarchical -filter "eco_status==eco_add"
```

If your design is congested, display the congestion map in the GUI and overlay the spare cell locations. Visually inspect the layout and identify the spare cells that are located in congested areas. You can prevent the tool from mapping to these spare cells by using the `set_attribute` command as shown in the following example:

```
icc_shell> set_attribute $spare_cells_in_congested_areas \
eco_change_status fs_dont_use
```

3. Perform a trial mapping of the ECO cells by using the `place_freeze_silicon -trial_map_spare_cells` command. This command moves the ECO cell to the target spare cell location, but does not map the ECO cell to the spare cell.
4. Analyze the results.

To analyze the results,

- Verify that all ECO cells have matching spare cells by running the `check_freeze_silicon` command.
- Verify that the design meets the QoR goals.
- Verify that the design meets the design rule constraints.
- If the design is congested, display the congestion map in the GUI and overlay the ECO cell locations. Visually inspect the layout and identify the ECO cells that are located in congested areas.

If all ECO cells have matching spare cells, the design meets the QoR goals, there are no DRC violations on ECO nets or cells, and there are no ECO cells placed in congested areas, go to step 6.

5. Finalize the ECO cell mapping by using the `place_freeze_silicon -map_spare_cells_only` command. This command maps each ECO cell to its target spare cell.
6. Update the power and ground connections by using the `derive_pg_connection` command.

7. For multivoltage designs, run the `check_mv_design` command and fix any violations. If the ECO changes are not compliant with the UPF constraints, edit the UPF constraints to match the ECO intent.
8. Update the routing by using the `route_zrt_eco` command or by manually routing the nets, as described in “[Performing ECO Routing](#).”

Hierarchical Placement Area Reservation ECO Flow

Use the hierarchical placement area reservation flow to add repeater cells on top-level nets over the existing soft macros without requiring design changes within the macro cells. The area that is reserved for the repeater cells is referred to as the reserved placement area. This flow lets you share placement resources over hierarchical cells, similar to sharing routing resources.

The hierarchical placement area reservation flow supports both top-down and bottom-up methodologies, as described in the following topics:

- [Top-Down Hierarchical Placement Area Reservation ECO Flow](#)
- [Bottom-Up Hierarchical Placement Area Reservation ECO Flow](#)

Top-Down Hierarchical Placement Area Reservation ECO Flow

In the top-down flow, you define the reserved placement area at the top level and then propagates that information down to the lower level.

The top-down hierarchical placement area reservation flow consists of the following steps:

1. At the top level, create placement blockages by using the `create_placement_blockage` command. The placement blockages are within the placement boundaries of the lower-level block. Set the `is_reserved_placement_area` attribute to `true` on the placement blockages by using the `set_attribute` command.
2. Derive the reserved placement area by using the `derive_reserved_placement_area -derive_type top_down` command.
3. Propagate the placement resources to the FRAM view by using the `create_macro_fram` command. This converts the reserved placement areas to placement sites.
4. Enable the site rows in the reserved placement areas for cell placement during legalization by setting the `legalize_enable_rpa_site_row` variable to `true`.
5. Insert repeater cells at the top level over the lower-level blocks by using the `add_buffer_on_route -no_eco_route` command, as described in “[Adding Repeater Cells on Routed Nets](#).”

6. Update the power and ground connections by using the `derive_pg_connection` command.
7. For multivoltage designs, run the `check_mv_design` command and fix any violations. If the ECO changes are not compliant with the UPF constraints, edit the UPF constraints to match the ECO intent.
8. Update the routing by using the `route_zrt_eco` command or by manually routing the nets, as described in “[Performing ECO Routing](#).”

Example 13-1 Script for Top-Down Hierarchical Placement Area Reservation

```
open_mw_cel postroute
create_placement_blockage -bbox {1000 1000 1860 1860} -name pb1
derive_reserved_placement_area -derive_type top_down
create_macro_fram -library_name macro.mw -cell_name mac1
set_app_var legalize_enable_rpa_site_row true
add_buffer_on_route -no_eco_route n999 BUFX1
derive_pg_connection
check_mv_design
route_zrt_eco
```

Bottom-Up Hierarchical Placement Area Reservation ECO Flow

In the bottom-up flow, you define the reserved placement area inside the lower-level block. When the lower-level block is completed and placed at the top level, the reserved placement area is available for inserting top-level repeater cells.

The bottom-up hierarchical placement area reservation flow consists of the following steps:

1. At the lower level, create placement blockages by using the `create_placement_blockage` command. Set the `is_reserved_placement_area` attribute to `true` on the placement blockages by using the `set_attribute` command.
2. At the top level, derive the reserved placement area by using the `derive_reserved_placement_area -derive_type bottom_up` command.
3. Propagate the placement resources to the FRAM view using the `create_macro_fram` command. This converts the reserved placement areas to placement sites.
4. Enable the site rows in the reserved placement areas for cell placement during legalization by setting the `legalize_enable_rpa_site_row` variable to `true`.
5. Insert repeater cells at the top level over the lower-level blocks by using the `add_buffer_on_route -no_eco_route` command, as described in “[Adding Repeater Cells on Routed Nets](#).”
6. Update the power and ground connections by using the `derive_pg_connection` command.

7. For multivoltage designs, run the `check_mv_design` command and fix any violations. If the ECO changes are not compliant with the UPF constraints, edit the UPF constraints to match the ECO intent.
8. Update the routing by using the `route_zrt_eco` command or by manually routing the nets, as described in “[Performing ECO Routing](#).”

Example 13-2 Script for Bottom-Up Hierarchical Placement Area Reservation Flow

```
# open soft macro CEL view
open_mw_cel mac1
create_placement_blockage -bbox {60 900 150 1000} -name pb1
set_attribute [get_placement_blockage pb1] \
    is_reserved_placement_area true
create_routing_blockage -layers {[get_layers -include_system \
    -filter {name=~metal*Blockage}]} -bbox {60 900 150 1000}

# open top-level CEL view
open_mw_cel postroute
derive_reserved_placement_area -derive_type bottom_up
create_macro_fram -library_name macro.mw -cell_name mac1
set_app_var legalize_enable_rpa_site_row true
add_buffer_on_route -no_eco_route n999 BUFX1
derive_pg_connection
check_mv_design
route_zrt_eco
```

Adding Spare Cells

You can add spare cells to a design by using one of the following methods:

- [Adding Spare Cells by Instantiating in the Verilog Netlist](#)
- [Adding Spare Cells by Using the `insert_spare_cells` Command](#)
- [Adding Programmable Spare Cells](#)

Adding Spare Cells by Instantiating in the Verilog Netlist

You can add spare cells to a design by manually instantiating them in the Verilog netlist, before you perform place and route.

Spare cells should be evenly distributed across the layout in anticipation of ECO changes at any location. To ensure even distribution, create multiple blocks containing spare cells and disperse them throughout the logic hierarchy.

To prevent the spare cells from creating noise and consuming power, tie their inputs to ground or power, as appropriate.

After you read in the Verilog netlist containing the instantiated spare cells, you must place them by using one of the following methods:

- [Automatically Placing Spare Cells During Placement](#)
- [Manually Placing Spare Cells Before Placement](#)
- [Manually Placing Spare Cells After Placement](#)

Automatically Placing Spare Cells During Placement

To automatically place the spare cells during placement and optimization,

1. Identify the spare cells by setting the `is_spare_cell` attribute on the spare cells with the `set_attribute` command.
2. Prevent optimization of the spare cells by setting the `dont_touch` attribute on the spare cells with the `set_dont_touch` command.
3. Run placement and optimization.
4. Allow for slight adjustments, if necessary, during subsequent optimization, by setting the `is_soft_fixed` attribute on the spare cells with the `set_attribute` command.

Example 13-3 Script to Automatically Place Spare Cells During Placement and Optimization

```
....  
set_attribute [get_cells *spares*] is_spare_cell true  
set_dont_touch [get_cells *spares*] true  
place_opt  
set_attribute [get_cells *spares*] is_soft_fixed true  
....
```

Manually Placing Spare Cells Before Placement

To manually place the spare cells before placement and optimization,

1. Identify the spare cells by setting the `is_spare_cell` attribute on the spare cells with the `set_attribute` command.
2. Place the spare cells in the desired area by using the `spread_spare_cells` command.
3. Prevent optimization of the spare cells by setting the `dont_touch` attribute on the spare cells with the `set_dont_touch` command.
4. Allow for slight adjustments, if necessary, during subsequent optimization, by setting the `is_soft_fixed` attribute on the spare cells with the `set_attribute` command.
5. Run placement and optimization.

Example 13-4 Script to Manually Place Spare Cells Before Placement and Optimization

```
....  
set_attribute [get_cells spares*] is_spare_cell true  
spread_spare_cells [get_cells spare*] -bbox {{5 5} {2000 2000}}  
set_dont_touch [get_cells spares*]  
set_attribute [get_cells spares*] is_soft_fixed true  
place_opt  
....
```

Manually Placing Spare Cells After Placement

To manually place the spare cells after placement and optimization,

1. Identify the spare cells by setting the `is_spare_cell` attribute on the spare cells with the `set_attribute` command.
2. Prevent placement and optimization of the spare cells by setting the `is_fixed` attributes with the `set_dont_touch_placement` command and a `dont_touch` attribute with the `set_dont_touch` command.
3. Run placement and optimization.
4. Enable the spare cells to be placed by removing the `is_fixed` attribute from the spare cells with the `remove_dont_touch_placement` command.
5. Place the spare cells in the desired area by using the `spread_spare_cells` command.
6. Run incremental legalization by using the `legalize_placement -incremental` command.
7. Allow for slight adjustments, if necessary, during subsequent optimization, by setting the `is_soft_fixed` attribute on the spare cells with the `set_attribute` command.

Example 13-5 Script to Manually Place Spare Cells After Placement and Optimization

```
....  
set_attribute [get_cells *spares*] is_spare_cell true  
set_dont_touch [get_cells *spares*] true  
set_dont_touch_placement [get_cells *spares*]  
place_opt  
remove_dont_touch_placement [get_cells spare*]  
spread_spare_cells [get_cells spare*] -bbox {{20 20} {999 999}}  
legalize_placement -incremental  
set_attribute [get_cells *spares*] is_soft_fixed true  
....
```

Adding Spare Cells by Using the `insert_spare_cells` Command

You can add and place spare cells to a legalized design by using the `insert_spare_cells` command after you perform placement and optimization.

To specify the library cells to use for the spare cell and the number of instances of each library cell, use the `-lib_cell` option with the `-num_instances` option. For example, to insert 250 instances each of the AND2 and OR2 library cells, use the following command:

```
icc_shell> insert_spare_cells -lib_cell {AND2 OR2} -num_instances 250
```

Alternatively, you can specify the library cells and a different number of instances for each library cell by using the `-num_cells` option. For example, to insert 200 instances of the NAND2 library cell and 150 instances of the NOR2 library cell, use the following command:

```
icc_shell> insert_spare_cells -num_cells {NAND2 200 NOR 150}
```

To insert the spare cells within a specific hierarchical block, use the `-hier_cell` option. When you use this option, the tool places the spare cells in a rectangular area that encloses all of the cells that belong to the specified hierarchical block so that the spare cells are close to those cells.

After the spare cells are inserted, set the following attributes on them by using the `set_attribute` command:

- The `dont_touch` attribute to prevent optimization
- The `is_soft_fixed` attribute to allow for slight adjustments, if necessary, during subsequent optimization

Example 13-6 Script to Insert Spare Cells Using the `insert_spare_cells` Command

```
place_opt
insert_spare_cells -lib_cell {INV} -num_instances 400 \
    -cell_name spares -tie
insert_spare_cells -lib_cell {AND2 OR2} -num_instances 200 \
    -cell_name spares 250 -tie
insert_spare_cells -lib_cell {XOR2} -num_instances 50 \
    -hier_cell MULT5 -cell_name spares -tie
set_attribute [get_cells spares*] is_soft_fixed true
set_dont_touch [get_cells spares*] true
```

Adding Programmable Spare Cells

The IC Compiler tool supports programmable spare cells, also known as gate array filler cells. During the ECO flow, the tool swaps an ECO cell with a programmable spare cell based on the cell type, cell width, and voltage area. If the tool removes an ECO cell from the design, it can reuse the corresponding programmable spare cell.

Insert programmable spare cells by using the `map_unit_tiles` command before you run the ECO flow. The `map_unit_tiles` command maps programmable spare cells in the library to unit tiles in the design. This mapping information is not persistent in the Milkyway design library, and is not saved when you save the design.

To report the mapping information, use the `report_unit_tiles` command.

Making ECO Changes

You can make ECO changes in the design by using one of the following two methods:

- [Importing a Verilog Netlist](#)
- [Using Netlist Editing Tcl Commands](#)

Importing a Verilog Netlist

You can make ECO changes to the design database by importing a Verilog netlist that includes the ECO changes with the `eco_netlist -by_verilog_file` command. If you are using the freeze silicon ECO flow or the freeze silicon QoR tuning ECO flow, you must use the `-freeze_silicon` option with the `eco_netlist -by_verilog_file` command.

The tool compares the current design to the input Verilog netlist and updates the current design based on the differences. However, by default, the tool ignores the following:

- Differences in the physical-only cells
 - To specify that the tool should consider the differences in the physical-only cells, use the `-physical` option.
- Differences in the power and ground connections
 - To specify that the tool should consider the differences in the power and ground connections, use the `-compare_pg` option.

To generate an output file containing the Tcl commands that correspond to the ECO changes, use the `eco_netlist -by_verilog_file -write_changes` command. You can edit this Tcl file and use it for subsequent ECO operations by using the `eco_netlist -by_tcl_file` command.

The tool can preprocesses the input Verilog netlist to prevent the following:

- The SYNOPSYS_UNCONNECTED nets that are created in the current design as a result of dangling bus pins in the input Verilog netlist.
- The tie-low connections that are created in the current design as a result of hierarchical dangling ports in the input Verilog netlist.

To enable this preprocessing, specify the following variable setting before you run the `eco_netlist -by_verilog_file` command:

```
icc_shell> set_app_var eco_netlist_preprocess_for_verilog true
```

Using Netlist Editing Tcl Commands

If the ECO changes are minimal, you can update the design by applying a Tcl file containing netlist editing commands with the `eco_netlist -by_tcl_file` command. If you are using the freeze silicon ECO flow or the freeze silicon QoR tuning ECO flow, you must use the `-freeze_silicon` option with the `eco_netlist -by_tcl_file` command.

Alternatively, you can run the netlist editing commands from the command line or apply the Tcl file by using the `source` command.

Use the netlist editing commands given in [Table 13-1](#) for the unconstrained ECO flow.

Table 13-1 Netlist Editing Commands for the Unconstrained ECO Flow

To do this	Use this command
Create a new port	<code>create_port</code>
Remove an existing port	<code>remove_port</code>
Create a new net	<code>create_net</code>
Remove an existing net	<code>remove_net</code>
Connect a net to a pin or port	<code>connect_net</code>
Disconnect a net from a pin or port	<code>disconnect_net</code>
Define a new bus	<code>define_bus</code>
Remove an existing bus definition	<code>undefine_bus</code>
Create a new cell	<code>create_cell</code>

Table 13-1 Netlist Editing Commands for the Unconstrained ECO Flow (Continued)

To do this	Use this command
Remove an existing cell	<code>remove_cell</code>
Resize a cell	<code>size_cell</code>
Insert a new buffer	<code>insert_buffer</code>
Remove an existing buffer	<code>remove_buffer</code>
To change the reference of an existing cell	<code>change_link</code> or <code>replace_cell_reference</code>

Use the netlist editing commands given in [Table 13-2](#) for the freeze silicon ECO flow, the freeze silicon QoR tuning ECO flow, or the freeze layer ECO flow.

Table 13-2 Netlist Editing Commands for the Freeze Silicon and Related ECO Flows

To do this	Use this command
Create a new cell	<code>create_cell -freeze_silicon</code>
Remove an existing cell	<code>remove_cell -freeze_silicon</code>
Resize a cell	<code>size_cell -freeze_silicon</code>
Insert a new buffer	<code>insert_buffer -freeze_silicon</code>
Remove an existing buffer	<code>remove_buffer -freeze_silicon</code>
To change the reference of an existing cell	<code>change_link -freeze_silicon</code>

When you use the `eco_netlist -by_tcl_file -freeze_silicon` command, the tool treats the netlist editing commands in the Tcl file as if the `-freeze_silicon` option is specified. As a result, all the related commands in the Tcl file are run under the freeze silicon mode.

Reverting Sized Cells

To revert the ECO changes made with the `size_cell` command, use the `revert_cell_sizing` command.

To include the sized cells that are adjacent to the specified cell in the same row, use the `-include_adjacent_sized_cells` option. When you use this option, by default, the tool also reverts the cells that abut the specified cell on either side, if they are also sized cells. However, if you use the `-adjacent_cells_distance` option, the tool recursively reverts the sized cells that are adjacent to sized cells within the specified distance in the same row.

Placing Cells During the Unconstrained ECO Flow

For the unconstrained ECO flow, place the ECO cells by using the `place_eco_cells` command.

This command derives the placement of each ECO cell based on the connectivity and the delays associated with the cell. The tool then legalizes each ECO cell to the closest unoccupied site. The existing cells are untouched to minimize the impact to their placement.

When you use the `place_eco_cells` command, specify the type of cells you want placed by using one of the following options:

- To place all unplaced cells, use the `-unplaced_cells` option.
- To place specific cells, use the `-cells` option.
- To place only the cells that have changed due to ECO operations, use the `-eco_changed_cells` option.

A cell is considered changed if the `eco_change_status` attribute of the cell has one of the following values:

- `create_cell`
- `change_link`
- `insert_buffer`
- `size_cell`

When you run the `eco_netlist` command, the tool automatically sets the `eco_change_status` attribute on the changed cells. If you update the design without using the `eco_netlist` command, you can manually set the `eco_change_status` attribute by using the `set_attribute` command.

By default, the `place_eco_cells` command places and legalizes the ECO cells. However, you can

- Perform placement only, and not legalization, by using the `-no_legalize` option
This option is mutually exclusive with the `-legalize_only` option.
- Perform legalization only, and not placement, by using the `-legalize_only` option
This option can be used only with the `-cells` or `-eco_changed_cells` option and is mutually exclusive with the `-no_legalize` option.

You can control legalization by specifying a displacement threshold by using the `-displacement_threshold` option. During legalization, if the displacement of an ECO cell exceeds this threshold, the tool does not legalize the ECO cell. In addition, the tool creates a collection named `epl_legalizer_rejected_cells` that consists of the ECO cells that are not legalized.

You can subsequently legalize the ECO cells in the `epl_legalizer_rejected_cells` collection by using the `legalize_placement -incremental -cells` command. This command moves existing cells to find legal locations for the ECO cells that do not have legal locations.

Placing and Mapping Cells During the Freeze Silicon ECO Flow

For the freeze silicon ECO flow or the freeze silicon QoR tuning ECO flow place and map the ECO cells to spare cells by using the `place_freeze_silicon` command.

This command places each ECO cell and maps it to the nearest matching spare cell. Cells that are deleted as a result of the ECO changes are not removed from the design. They are converted to spare cells and remain in the design.

To map an ECO cell to a specific spare cell, use the `map_freeze_silicon` command. To map multiple ECO cells to specific spare cells, use the `map_freeze_silicon -map_file` command.

The mapping file has the following format:

```
cell_1    spare_cell_1
cell_2    spare_cell_2
...
...
```

Adding Repeater Cells on Routed Nets

To add repeater cells on a fully routed net, use the `add_buffer_on_route` command.

When you use this command, you must specify the following information:

- The net on which to add the repeater cells
- The library cell to use for the repeater cells

By default, the tool assumes that the library cell you specify is a buffer. To add pairs of inverters as repeater cells instead of buffers, use the `-inverter_pair` option.

The following example shows how to add pairs of inverters as repeater cells on the net4 net using the mylib/INV1 library cell.

```
icc_shell> add_buffer_on_route net4 [get_lib_cells mylib/INV1] \
    -inverter_pair -repeater_distance 100
```

- The location to add the repeater cells

You specify the repeater cell locations by using one of the following methods:

- Exact locations

To specify the locations of the repeater cells, use the `-location` option. When you specify an exact location, you can also specify a layer with an existing routing shape at that location to connect the repeater cell to.

The following example adds the mylib/BUF1 library cell as a repeater cell on the net1 net at location (100, 200) and connects it to an existing route shape on the M4 layer.

```
icc_shell> add_buffer_on_route net1 [get_lib_cells mylib/BUF1] \
    -location {100 200 M4}
```

- Fixed intervals

To specify the distance in microns between the repeater cells, use the `-repeater_distance` option. To specify the distance in microns between the driver of the net and the first repeater cell, use the `-first_distance` option.

The following example adds the mylib/BUF1 library cell as a repeater cell on the net2 net at an interval of 150 microns. The distance between the driver and the first repeater cell is 100 microns.

```
icc_shell> add_buffer_on_route net2 [get_lib_cells mylib/BUF1] \
    -first_distance 100 -repeater_distance 150
```

- Relative intervals

To specify the distance between the repeater cells as a ratio of the total net length, use the `-repeater_distance_length_ratio` option. To specify the distance

between the driver and the first repeater as a ratio of the total net length, use the `-first_distance_length_ratio` option.

The following example adds the mylib/BUF1 library cell as a repeater cell on the net3 net at an interval that is 20 percent of the total net length. The distance between the driver and the first repeater cell is 10 percent of the total net length.

```
icc_shell> add_buffer_on_route net3 [get_lib_cells mylib/BUF1] \
    -first_distance_length_ratio 0.1 \
    -repeater_distance_length_ratio 0.2
```

Instead of specifying the library cells to use and their locations as exact locations, fixed intervals, or relative intervals, you can specify an exact configuration of repeater cells to insert by using the `-user_specified_buffers` option.

When you use the `-user_specified_buffers` option, for each repeater cell, you must specify the instance name, the library cell to use, the x- and y-coordinates of the exact location, and a layer at that location with an existing routing shape to connect to. You can insert different types of buffers or inverter pairs by using this option, but you cannot combine both buffers and inverter pairs.

The following example adds two repeater cells named ECO1 and ECO2 on net n22. The ECO1 cell is of type BUF2, at location (100, 70), connecting to a routing shape on the M3 layer. The ECO2 cell is of type BUF4, at location (150, 70), also connecting to a routing shape on the M3 layer.

```
icc_shell> add_buffer_on_route net22 \
    -user_specified_buffers {ECO1 BUF2 100 70 M3 ECO2 BUF4 150 70 M3}
```

By default, the `add_buffer_on_route` command

- Adds repeater cells over placement blockages, soft macros, and hard macros.
To prevent this, use the `-respect_blockages` option.
- Adds repeater cells on both global routed and detail routed nets.
To add cells only on nets that are global routed only, without assigned tracks or detail routing, use the `-only_global_routed_nets` option.
- Does not add repeater cells on a route segment if it is necessary to create new ports because of a difference in the logical and physical topology of the net.
To add repeaters on such route segments by creating new ports, use the `-punch_port` option.

- Legalizes the repeater cells added.

To control the distance a repeater cell is moved during legalization, use the `-max_distance_for_legalized_location` option. If the tool cannot legalize a repeater cell within the specified distance, it is not added.

To disable legalization of the repeater cells, use the `-no_legalize` option. If you disable legalization, the affected nets are not rerouted.

- Reroutes the affected nets by following the original routes as much as possible.

To disable rerouting of the affected nets, use the `-no_eco_route` option.

Performing ECO Routing

If the changes are minimal and is limited to a few layers, you can manually route the nets changed by the ECO process.

When the change is large, use the `route_zrt_eco` command to route the affected nets. This command routes all open nets and newly added nets, and cleans up the deleted nets, obsolete nets, and unused sections of changed wires.

By default, the `route_zrt_eco` command can reroute any noncritical net to fix routing DRC violations. As a result, nets that are unchanged by the ECO process might be rerouted. This default behavior is suitable when there are many ECO changes.

To control the rerouting of nets unchanged by the ECO process, use the following option settings with the `route_zrt_eco` command:

- To route the ECO nets first and reroute the other nets only for fixing routing DRC violations, use the `-reroute modified_nets_first_then_others` option.

This reduces the layout changes and is suitable when the there are only a small number of ECO changes.

- To route only the ECO nets, use the `-reroute modified_nets` option.

This is very restrictive and can result in the tool being unable to fix all routing DRC violations. As a result, it should be used only when there are only a very few ECO changes.

After performing ECO routing, use the `extract_rc -incremental` command to perform incremental extraction on the changed nets. The tool runs the incremental extraction based on the previous extraction results in the same session.

Limits ECO Routing to Specific Layers

You can reduce the cost of the layout masks by limiting the routing layers used during the ECO routing.

Before ECO routing, prepare the net shapes and vias by performing the following tasks:

1. Query the floating net shapes and vias of any removed nets by using the `get_net_shapes` and `get_vias` commands.
2. Remove the selected floating net shapes and vias on particular layers by using the `remove_user_shape` and `remove_via` commands.
3. Manually reassign floating net shapes to nets that need to be routed or rerouted by using the `set_attribute` command.

Limit the ECO routing to specific metal layers by using the `set_route_zrt_common_options` command as follows:

- To specify the layers that are frozen during ECO routing, use the `-freeze_layer_by_layer_name` option.
- To specify the layers on which the router can remove, but not add, routes, use the `-forbid_new_metal_by_layer_name` option.
- To control the treatment of vias that touch frozen layers, use the `-freeze_via_to_frozen_layer_by_layer_name` option.

Reporting ECO History and Physical Changes

To generate a report of the ECO changes, use the `report_eco_history` command. By default, the `report_eco_history` command outputs a change list of the latest ECO changes to the IC Compiler log file. You can save the output to a Tcl script file instead by using the `-output` option. You can also report a summary of all ECO changes made to the design by using the `-summary` option.

To report a summary of the ECO cell displacement and ECO net length changes, use the `report_eco_physical_changes` command.

To report detailed information about the ECO cell displacement and ECO net length changes, use the `-type` option with the `report_eco_physical_changes` command as follows:

- To report detailed information about the ECO cell displacement, use the `-type cell_displacement` setting.
- To report detailed information about the ECO net length changes, use the `-type net_length` setting.
- To report detailed information about both the ECO cell displacement and the ECO net length changes, use the `-type all` setting.

Comparing Designs

You can use the `eco_netlist` command with the `-by_design` and `-working_design` options to analyze the logical differences in a design at different stages of the design flow. For example,

- Before and after optimization
- Before and after clock tree synthesis
- Before and after ECO

With these options, the `eco_netlist` command compares the working design netlist to the golden design that is in the same or a different design library. The working design can be your current design or a design you specify.

When comparing designs after ECO, either the `-by_verilog_file`, `-by_design`, or `-by_tcl_file` options is required; these options are mutually exclusive.

14

In-Design Signoff Timing Analysis and Correlation Checking

This chapter describes how to reduce timing violations during final signoff by using In-Design signoff timing analysis and correlation checking.

The IC Compiler tool is an implementation engine, and the StarRC and PrimeTime tools are signoff engines. In general, the implementation engine and the signoff engine are well correlated with each other. However, small variations can occur because

- The PrimeTime and StarRC tools, as signoff tools, are more accurate.
- The PrimeTime tool has reduced signal integrity pessimism.
- The IC Compiler tool might have implementation margins.
- The IC Compiler tool might have fewer corners.

These differences can cause timing violations during final signoff. To reduce violations, you should use In-Design signoff timing analysis and correlation checking.

This chapter describes In-Design signoff timing analysis and correlation checking in the following sections:

- [In-Design Signoff Timing Analysis](#)
- [Checking the Signoff Correlation](#)

In-Design Signoff Timing Analysis

In-Design signoff timing analysis uses the `run_signoff` command to run signoff timing analysis. After you run the `run_signoff` command, you can run PrimeTime commands in the IC Compiler tool to analyze the timing based on signoff timing. You can also perform manual optimization to fix violations reported during analysis. For example, you manually edit the netlist to do different optimizations, such as buffer insertion, buffer removal, or cell sizing. Both the IC Compiler and PrimeTime tools simultaneously update the netlist with your changes, and the PrimeTime tool runs an incremental timing update to verify the effects of the changes.

In-Design signoff timing analysis is described in the following sections:

- [Key Features](#)
 - [Running In-Design Signoff Timing Analysis](#)
 - [Correlating Multicorner-Multimode Scenarios](#)
 - [Enabling On-Chip Variation](#)
 - [Enabling Static Noise Analysis](#)
 - [Setting Up the PrimeTime Tool](#)
 - [Setting Up the StarRC Tool](#)
 - [Automatically Generating the PrimeTime and StarRC Settings](#)
 - [Running the `run_signoff` Command](#)
 - [Previewing Slack Improvements](#)
 - [Troubleshooting and Problem Solving](#)
-

Key Features

In-Design signoff timing analysis supports the following key features:

- Multicorner-multimode support

In-Design signoff timing analysis supports multicorner-multimode designs. If you have defined more than one scenario for your design, the IC Compiler tool runs PrimeTime distributed multi-scenario analysis and multiple StarRC sessions, one for each parasitic corner. In the PrimeTime tool, a unique extraction corner requires a technology file plus an operating temperature. This is similar to how the IC Compiler tool defines a corner by using a TLUPlus file plus an operating temperature.

- IEEE 1801 Unified Power Format (UPF) support

In-Design signoff timing analysis supports designs with UPF constraints. IC Compiler automatically passes the UPF constraints to the PrimeTime tool when you run signoff analysis.

- Automatic generation of PrimeTime and StarRC settings

In-Design signoff timing analysis runs PrimeTime and StarRC signoff tools to find timing violations. The IC Compiler tool can automatically generate the necessary PrimeTime and StarRC settings for signoff timing analysis.

For more information, see [“Automatically Generating the PrimeTime and StarRC Settings” on page 14-18](#).

- On-chip variation (OCV) and advanced on-chip variation (AOCV) modeling support
- Path-based analysis support
- Distributed multi-scenario analysis support
- Bottleneck analysis support

In-Design signoff timing analysis supports the PrimeTime bottleneck analysis commands, including `report_bottleneck` and `report_si_bottleneck`. These commands are available after running signoff analysis.

Running In-Design Signoff Timing Analysis

To run In-Design signoff timing analysis,

1. If you have a multicorner-multimode design, ensure that the scenarios are defined and the correct set of scenarios is set as active.

All active scenarios in the IC Compiler tool are analyzed in PrimeTime distributed multi-scenario analysis. Inactive scenarios in the IC Compiler tool are not analyzed in the PrimeTime tool.

For information about defining scenarios and checking the scenario definitions in IC Compiler, see [“Setting Up for Multicorner-Multimode Analysis and Optimization” on page 3-33](#). For information about correlating scenarios between the IC Compiler and PrimeTime tools, see [“Correlating Multicorner-Multimode Scenarios” on page 14-7](#).

2. If desired, enable OCV and static noise analysis.

For information about enabling OCV analysis, see [“Enabling On-Chip Variation” on page 14-9](#). For information about enabling static noise analysis, see [“Enabling Static Noise Analysis” on page 14-9](#).

3. Set up the signoff tool environment.

To set up the signoff tool environment, you must specify the locations of the PrimeTime and StarRC executables, as well as define any custom settings required for running these tools. For information about setting up the PrimeTime tool, see “[Setting Up the PrimeTime Tool](#)” on page 14-10. For information about setting up the StarRC tool, see “[Setting Up the StarRC Tool](#)” on page 14-15.

4. Perform signoff timing analysis by running the `run_signoff` command.

You can also use the GUI by choosing Signoff > Analysis and selecting the full analysis option. After you run this command, the IC Compiler tool enters signoff analysis mode, and the PrimeTime tool is enabled as the timing engine.

For details about running the `run_signoff` command, see “[Running the run_signoff Command](#)” on page 14-20

5. Review any violations.

Use the `report_timing` and `report_constraint` commands as needed.

6. Manually fix the violations.

Use the `get_alternative_lib_cells`, `size_cell`, `insert_buffer`, and `remove_buffer` commands as needed. You can run these commands from the GUI ECO menu. Other netlist editing commands, such as `create_cell`, are not supported by In-Design signoff timing analysis. From the Size Cell dialog box, you can preview slack improvement. For details, see “[Previewing Slack Improvements](#)” on page 14-23.

7. Run incremental legalization and ECO routing.

Use the `place_eco_cells -legalize_only` command to run incremental legalization, which legalizes any newly added cells to legal locations. Use the `route_zrt_eco` command to run ECO routing.

8. Perform incremental signoff analysis by running the `run_signoff -incremental` command.

You can also use the GUI by choosing Signoff > Analysis and selecting the incremental analysis option.

9. Review any violations. Repeat steps 6 through 9 until all violations are fixed.

After you have completed In-Design signoff timing analysis, disable signoff analysis mode by using the `run_signoff -signoff_analysis false` command. When you disable signoff analysis mode, you exit the PrimeTime timing engine and return to the IC Compiler timing engine.

[Example 14-1](#) shows an example script for running the `run_signoff` command.

Example 14-1 Using the run_signoff Command

```
# Set up the In-Design signoff timing analysis environment
set pt_dir /tools/primetime/G-2012.06/amd64/syn/bin
set star_dir /tools/star_rcxt/G-2012.06/amd64_star-rcxt/bin
set_starrcxt_options -exec_dir $star_dir -map_file $MAP \
    -max_nxtgrd_file $MAX_GRD -min_nxtgrd_file $MIN_GRD
set_primestime_options -exec_dir $pt_dir -sdc_file $SDC
report_starrcxt_options
report_primestime_options

# Perform signoff analysis
run_signoff

# Review the violations
report_qor
report_timing

# Fix the violations
size_cell ...
insert_buffer ...

# Run incremental legalization and ECO routing
place_eco_cells -unplaced_cells
route_zrt_eco

# Perform incremental signoff analysis
run_signoff -incremental

# Review the violations
report_qor
report_timing
```

[Example 14-2](#) shows an example script for a multicorner-multimode design.

Example 14-2 Using the run_signoff Command With Multiple Scenarios

```
set pt_dir /tools/primetime/F-2011.06/amd64/syn/bin
set star_dir /tools/star_rcxt/F-2011.06/amd64_star-rcxt/bin

open_mw_cel post_route_opt

# Set up the In-Design signoff timing analysis environment
set_primestime_options -exec_dir $pt_dir
set_starrcxt_options -exec_dir $star_dir -map_file $map_file
report_primestime_options
report_starrcxt_options
set_host_options -target PrimeTime -pool lsf
report_host_options
```

```
# Create three scenarios with scenario-specific settings
create_scenario maxmax
    set_operating_conditions -analysis_type on_chip_variation Worst
    set_tlu_plus_files -max_tluplus $max_tlu_file -tech2itf_map $map_file
    set_primetetime_options -sdc_file $max_sdc_file
    set_starrcxt_options -max_nxtgrd_file $max_grd_file

create_scenario minmin
    set_operating_conditions -analysis_type on_chip_variation Best
    set_tlu_plus_files -max_tluplus $min_tlu_file -tech2itf_map $map_file
    set_primetetime_options -sdc_file $min_sdc_file
    set_starrcxt_options -max_nxtgrd_file $min_grd_file
create_scenario nomnom
    set_operating_conditions -analysis_type on_chip_variation Nom
    set_tlu_plus_files -max_tluplus $nom_tlu_file -tech2itf_map $map_file
    set_primetetime_options -sdc_file $nom_sdc_file
    set_starrcxt_options -max_nxtgrd_file $nom_grd_file

# Perform signoff analysis
run_signoff

# Review the violations
report_timing
report_constraint -all_violators

# Fix the violations
get_alternative_lib_cells
size_cell
insert_buffer

# Run incremental legalization and ECO routing
place_eco_cells -legalize_only
route_zrt_eco

# Perform incremental signoff analysis
run_signoff -incremental

# Review the violations
report_timing
report_qor

# Disable signoff analysis mode
run_signoff -signoff_analysis false

save_mw_cel -as final
```

Correlating Multicorner-Multimode Scenarios

When creating scripts for In-Design signoff timing analysis, you need to understand how the IC Compiler and PrimeTime tools handle scenarios differently. Both tools define scenarios the same way by using the `create_scenario` command, but each tool uses a different command to specify the scenarios to be used in the current run. The IC Compiler tool uses the `set_active_scenarios` command and the PrimeTime tool uses the `current_session` command. Keep the following in mind:

- The scenario concept is the same for both the IC Compiler and PrimeTime tools.
- The actual user interface commands have differences.
- The IC Compiler tool's *active scenario* is similar to PrimeTime's *session*.
- The IC Compiler tool does not support the PrimeTime `current_session` command. You need to use the `set_active_scenarios` command instead.
- Each PrimeTime process reads only one corner for the `read_parasitics` command.
- The IC Compiler tool reads multiple corners for the `read_parasitics` command.

After you run the IC Compiler `current_scenario` command, most reporting commands and the `read_parasitics` command are applied only to the specified current scenario. All other commands, such as optimization and netlist editing commands, are applied to all scenarios. The IC Compiler `current_scenario` command supports only a single scenario and does not support the `-all` option used by the PrimeTime `current_scenario` command.

After you run the PrimeTime `current_scenario` command, all commands are applied to the specified current scenario. Even optimization commands, such as `size_cell`, change only the given scenarios. The PrimeTime `current_scenario` command supports multiple scenarios by using the following options:

- `current_scenario -all`
- `current_scenario {Func Test}`

After you run the IC Compiler `run_signoff` command, the `current_scenario` behavior is changed to update both the IC Compiler and PrimeTime tools.

- `current_scenario -all`

No change occurs to the current scenario.

Command focus is changed to include all scenarios in PrimeTime distributed multi-scenario analysis.

- `current_scenario Func`

Running the `current_scenario Func` command changes the current scenario to Func.

Running the `current_scenario Func` command changes the command focus to Func in PrimeTime distributed multi-scenario analysis.

Mapping Scenarios Between the IC Compiler and PrimeTime Tools

There must be a one-to-one mapping between active scenarios in the IC Compiler tool to scenarios in the PrimeTime tool. Because the IC Compiler tool does not split a scenario based on minimum and maximum conditions automatically, you must redefine all your best-case and worst-case scenarios.

For example, assume the following best-case and worst-case scenario in IC Compiler multicorner-multimode:

```
set_tlu_plus_files -max_tluplus max.tlup -min_tluplus min.tlup
set_min_library -min_version fast.db slow.db
set_operating_conditions -analysis_type bc_wc
```

When you use the `set_operating_conditions -analysis_type bc_wc` command, you can specify different values for the `-max_cap_scale` and `-min_cap_scale` options of the `set_extraction_options` command for the two different corners.

When you run In-Design signoff timing analysis, you must replace the single scenario with two scenarios:

- Worst-case scenario

```
set_tlu_plus_files -max_tluplus max.tlup
# Use slow.db as the link_library
set_operating_conditions -analysis_type on_chip_variation
```

- Best-case scenario

```
set_tlu_plus_files -max_tluplus min.tlup
# Use fast.db as the link_library
set_operating_conditions -analysis_type on_chip_variation
```

For each scenario, you must use on-chip variation (OCV) and you cannot use `set_min_library`, `minimum TLUPlus`, `minimum nxtgrd`, `minimum images` for the PrimeTime tool, or `minimum images` for the StarRC tool.

Enabling On-Chip Variation

If you enable crosstalk analysis, OCV is automatically supported. If you do not enable crosstalk analysis, you need to enable OCV analysis by using the following command:

```
icc_shell> set_operating_conditions -analysis_type on_chip_variation
```

When you use the `set_operating_conditions -analysis_type on_chip_variation` command, you must specify the same value for both the `-max_cap_scale` and `-min_cap_scale` options of the `set_extraction_options` command, as they both apply to the same corner.

Enabling Static Noise Analysis

To improve QoR, you can enable the tool to report static noise violations by running the `set_si_options -static_noise true` command. When you use this command, the following commands are based on the PrimeTime tool after you run the `run_signoff` command:

- `report_noise`
- `report_noise_parameters`
- `report_noiseViolation_sources`
- `report_si_aggressor_exclusion`
- `report_si_bottleneck`
- `report_si_double_switching`
- `report_si_delay_analysis`
- `report_si_noise_analysis`

Setting Up the PrimeTime Tool

Before you run In-Design signoff timing analysis, you must specify the location of the PrimeTime executable by using the `-exec_dir` option with the `set_primate_options` command. This location must contain the executable for PrimeTime version G-2012.06 or later.

By default, when the IC Compiler tool runs the PrimeTime tool for In-Design signoff timing analysis, it uses the default options of PrimeTime and the PrimeTime SI default filters, and does not use distributed processing. The following sections describe how to modify these settings:

- [Modifying the PrimeTime Settings](#)
- [Enabling Consistency Settings](#)
- [Configuring PrimeTime Distributed Processing](#)

Modifying the PrimeTime Settings

You can use the `set_primate_options` command to define PrimeTime settings to be used in the flow. You can define the following settings:

- The SDC file
 - To specify the SDC file used in the PrimeTime runs, use the `-sdc_file` option. For multicorner-multimode designs, the `-sdc_file` option is scenario-specific and applies only to the current scenario; you specify one SDC file per scenario.
- The saved sessions
 - To create a saved session in the PrimeTime tool, use the `save_session` command.
 - To use a saved session of the PrimeTime tool, specify the file name of the PrimeTime saved session by using the `-max_image` or `-min_image` option with the `set_primate_options` command.
 - For non-multicorner-multimode designs, you can specify a minimum and a maximum saved session. To specify the minimum saved session, use the `-min_image` option. To specify the maximum saved session, use the `-max_image` option.
 - For multicorner-multimode designs, you can specify only maximum saved sessions. The maximum saved session setting is scenario-specific and applies only to the current scenario; you specify one file per scenario.

Note:

Minimum saved sessions are not supported for multicorner-multimode designs.

- The common setup file

The common setup file is used for all PrimeTime runs, including both best-case, worst-case analysis and multicorner-multimode analysis. This file is sourced into the PrimeTime tool before loading the netlist and is used to set up Tcl variables. To specify this file, use the `-common_file` option with the `set_primate_options` command.

Only the following variables are allowed in the common setup file:

- All RC variables
- All case analysis variables
- All CCS variables
- All path-based analysis variables
- All report variables
- All timing variables except `timing_save_pin_arrival_and_slack`
- All signal integrity variables except `si_enable_analysis`
- All parasitic variables except `read_parasitics_load_locations`
- All variation variables except `variation_enable_analysis` and `variation_derived_scalar_attribute_mode`

If the file contains any other variables, the tool reports a PSYN-387 error.

- The run-specific command file

The run-specific command file is used for a specific PrimeTime run, such as best-case analysis, worst-case analysis, or scenario-specific analysis. This file is sourced into the PrimeTime tool after loading the netlist and is used to specify the analysis commands. To specify these files, use the `-specific_file` option with the `set_primate_options` command.

For non-multicorner-multimode designs, you can specify up to two files in best-case, worst-case mode, one for worst-case analysis and one for best-case analysis. You specify the files using the following format: `{max_file min_file}`. If you specify only one file, the same file is used for both worst-case and best-case analysis.

For multicorner-multimode designs, this option is scenario-specific and applies only to the current scenario; you specify one file per scenario.

Only the following commands are allowed in the run-specific command file:

- All report commands
- All check commands
- All update commands

- All set commands except `set_units` and `set_unix_variable`
- All reset commands except `reset_design`
- `create_correlation` and `create_variation`
- `define_scaling_lib_group`
- `group_path`
- `read_aocvm` and `remove_aocvm`
- `scale_parasitics` and `complete_net_parasitics`

If the file contains any other commands, the tool reports a PSYN-387 error.

[Example 14-3](#) shows an example of using the `set_primateime_options` command.

Example 14-3 Setting PrimeTime Options

```
set SDC pt.sdc
set pt_dir /tools/primetime/G-2012.06/amd64/syn/bin
set_primateime_options -exec_dir $pt_dir -sdc_file $SDC \
    -max_image pt_max \
    -min_image pt_min
```

To verify your PrimeTime option settings, use the `report_primateime_options` command.

[Example 14-4](#) shows an example report generated by the `report_primateime_options` command.

Example 14-4 PrimeTime Options Report

```
icc_shell> report_primateime_options
*****
Report : PrimeTime_options
Design : tutorial
Version:
Date   :
*****
PrimeTime exec dir: / tools/primetime/G-2012.06/amd64/syn/bin
PrimeTime SDC file: /home/signoff/tutorial/pt.sdc
PrimeTime max image: pt_max
PrimeTime min image: pt_min
```

After you specify your PrimeTime settings by using the `set_primateime_options` command, you can check for consistency between the IC Compiler and PrimeTime variable settings by using the `check_primateime_icc_consistency_settings` command.

Consistent settings ensure good timing correlation between the two tools. The command reports an error for each variable setting that is inconsistent between the two tools and tells

you the proper setting for the variable in the applicable tool, as shown in the following example:

```
icc_shell> check_primestime_icc_consistency_settings
Error: The auto_wire_load_selection setting is inconsistent between
IC Compiler and PrimeTime. The IC Compiler value is FALSE; the
PrimeTime value is "true". (PSYN-803)
Error: The auto_wire_load_selection IC Compiler variable should be set
to TRUE. (PSYN-802)
```

Enabling Consistency Settings

To automatically change certain IC Compiler and PrimeTime settings to values recommended for consistency, use the `enable_primestime_icc_consistency_settings` command. When you run this command, it reports the changes that it makes.

For example,

```
icc_shell> enable_primestime_icc_consistency_settings
set ignore_clock_input_delay_for_skew false
# ignore_clock_input_delay_for_skew changed to true
set timing_edge_specific_source_latency false
# timing_edge_specific_source_latency changed to true
set timing_enable_multiple_clocks_per_reg false
# timing_enable_multiple_clocks_per_reg changed to true
```

By default, this command does not modify settings that enable a specific flow, such as AOCV or signal integrity, and variables that are specific to either the IC Compiler or PrimeTime tool. To modify all settings, use the `-all` option.

To report the changes without actually implementing them, use the `-report_recommended` option. To save the original settings in a file, use the `-original_variable_file` option.

Configuring PrimeTime Distributed Processing

By default, In-Design signoff timing analysis uses the remote shell (`rsh`) program to invoke the PrimeTime processes. To use the C-shell (`csh`) instead,

1. Create a `runcmd` file as follows:

```
#!/bin/csh -f
$argv &
```

2. Use the `set_host_options -target PrimeTime` command to configure the PrimeTime tool to use this `runcmd` file.

```
icc_shell> set_host_options -target PrimeTime -submit_command runcmd
```

In-Design signoff timing analysis can use distributed processing to perform PrimeTime timing analysis for multicorner-multimode designs. Use the `set_host_options -target PrimeTime` command to define the distributed processing configuration. If you use the

set_host_options -target PrimeTime command multiple times, the number of hosts is cumulative. For example, if you run the following commands:

```
icc_shell> set_host_options -pool lsf -num_processes 4
icc_shell> set_host_options -num_processes 2 unix1
icc_shell> set_host_options -num_processes 2 unix2
```

The IC Compiler tool runs the first four processes on LSF hosts, the next two processes on the machine named unix1, and the last two processes on the machine named unix2. Note that scenarios are not run on the local host unless explicitly defined in the set_host_options command.

In-Design signoff timing analysis starts running PrimeTime distributed processes as soon as a single host becomes available. Other hosts start PrimeTime processes as soon as they become available. Unlike PrimeTime distributed multi-scenario analysis, which allows you to specify how many hosts must be available before you can run another task, the IC Compiler tool always sets this value to one. If the number of scenarios is more than the number of hosts specified by the set_host_options command, the tool waits until a host becomes free to run the additional scenarios; this behavior is the same as the PrimeTime distributed multi-scenario analysis behavior.

For example, if you have 15 scenarios and specify only 5 hosts by using the following command:

```
icc_shell> set_host_options -pool lsf -num_processes 5
```

the tool analyzes the 15 scenarios using the 5 LSF hosts, based on a daisy-chain schedule.

Note that if you use the Oracle Grid Engine for your PrimeTime distributed processes, you must submit the job as a binary job by using the -b y option; otherwise, the PrimeTime tool fails. The following example uses the -b y option.

```
icc_shell> set_host_options -target PrimeTime -pool grd \
    -num_processes 8 \
    -submit_options {-cwd -P bnormal -b y -l arch=glinux,cputype=amd64}
```

These options direct the qsub command to run the specified script directly without making an intermediate copy, which speeds up the submission of the jobs. For details, see the qsub man page.

For details about using the set_host_options command to define a distributed processing configuration, see “[Configuring Distributed Processing](#)” on page 2-14.

To verify the host option settings used for the PrimeTime tool, use the report_host_options -target PrimeTime command.

Setting Up the StarRC Tool

Before you run In-Design signoff timing analysis, you must specify the location of the StarRC executable by using the `-exec_dir` option with the `set_starrcxt_options` command. This location must contain the executable for StarRC version G-2012.06 or later.

By default, when the IC Compiler tool runs the StarRC tool for In-Design signoff timing analysis, it uses the following StarRC option settings:

- MODE: 200
- OPERATING_TEMPERATURE: 25
- REDUCTION: TOPOLOGICAL

Note:

This setting provides the best QoR; however, to achieve better correlation with the standalone signoff tools, use a StarRC option file to set the `REDUCTION` option to `YES`. For information about using a StarRC option file, see the next section, “[Modifying the StarRC Settings](#).”

In addition, it does not use distributed processing for the StarRC runs. The following sections describe how to modify these settings.

Modifying the StarRC Settings

You can use the `set_starrcxt_options` command to define StarRC settings to be used in the flow. You can define the following settings:

- The StarRC layer mapping file

The layer mapping file, which maps database layers to TCAD GRD layers, is needed for every StarRC run. To specify the layer mapping file, use the `-map_file` option. For details about the layer mapping file, see the *StarRC User Guide and Command Reference*.

- The nxtgrd files

For non-multicorner-multimode designs, you can specify a minimum and a maximum nxtgrd file. To specify the minimum nxtgrd file, use the `-min_nxtgrd_file` option. To specify the maximum nxtgrd file, use the `-max_nxtgrd_file` option.

For multicorner-multimode designs, you can specify only maximum nxtgrd files. The maximum nxtgrd file setting is scenario-specific and applies only to the current scenario; you must set it for each active scenario for your design.

Note:

Minimum nxtgrd files are not supported for multicorner-multimode designs.

- The saved sessions

To use a saved session of the StarRC tool, specify the name of the run directory used by the session by using the `-max_image` or `-min_image` option with the `set_starrcxt_options` command.

For non-multicorner-multimode designs, you can specify a minimum and a maximum saved session. To specify the minimum saved session, use the `-min_image` option. To specify the maximum saved session, use the `-max_image` option.

For multicorner-multimode designs, you can specify only maximum saved sessions. The maximum saved session setting is scenario-specific and applies only to the current scenario; you must set it for each active scenario for your design.

Note:

Minimum saved sessions are not supported for multicorner-multimode designs.

- The StarRC mode

You can specify a StarRC mode by using the `-mode` option with the `set_starrcxt_options` command. The valid values are 200 and 400. The default is 200. The mode establishes the tradeoff between runtime and accuracy. For more information, see the *StarRC User Guide and Command Reference*.

You can also set the mode by setting the `MODE` option in the StarRC options file.

- The StarRC options file

The StarRC options file contains option settings used for all StarRC runs. To specify this file, use the `-option_file` option with the `set_starrcxt_options` command.

The following options are reserved and are automatically set by the IC Compiler tool; these options cannot appear in the StarRC option file:

- BLOCK
- COUPLE_TO_GROUND
- EXTRA_GEOMETRY_INFO
- FSCOMPARE_FILE_PREFIX
- FS_PREPROCESSOR_STAGE
- HIERARCHICAL_SEPARATOR
- INCREMENTAL
- MAPPING_FILE
- MILKYWAY_DATABASE
- MILKYWAY_EXPAND_HIERARCHICAL_CELLS

- MULTI_CPU_ON_LICENSE_ONLY
- NETLIST_FILE
- NETLIST_FORMAT
- NETLIST_INCREMENTAL
- NETLIST_NODE_SECTION
- NETLIST_TAIL_COMMENTS
- STAR_DIRECTORY
- SENSITIVITY
- TARGET_ANALYSIS
- TCAD_GRD_FILE

If the file contains any of these commands, the tool reports a PSYN-171 error.

[Example 14-5](#) shows an example of using the `set_starrcxt_options` command.

Example 14-5 Setting StarRC Options

```
set star_dir /tools/star_rcxt/G-2012.06/amd64_star-rcxt/bin
set_starrcxt_options -exec_dir $star_dir \
    -max_nxtgrd_file $MAX_GRD \
    -min_nxtgrd_file $MIN_GRD \
    -max_image star_max \
    -min_image star_min \
    -map_file $MAP
```

To verify your StarRC option settings, use the `report_starrcxt_options` command.

[Example 14-6](#) shows an example report generated by the `report_starrcxt_options` command.

Example 14-6 StarRC Options Report

```
icc_shell> report_starrcxt_options
*****
Report : StarRCXT_Options
Design : tutorial
Version:
Date   :
*****
StarRCXT exec dir: /tools/star_rcxt/G-2012.06/amd64_star-rcxt/bin
StarRCXT max nxtgrd file: /libraries/7lm.nxtgrd
StarRCXT min nxtgrd file: /libraries/7lm.nxtgrd
StarRCXT map file: /libraries/90_7lm.map
StarRCXT max image: star_max
StarRCXT min image: star_min
```

Configuring StarRC Distributed Processing

By default, In-Design signoff timing analysis uses the remote shell (rsh) program to invoke the StarRC processes. To use the C-shell (csh) instead,

1. Create a runcmd file as follows:

```
#!/bin/csh -f  
$argv &
```

2. Use the `set_host_options -target StarRC` command to configure the StarRC tool to use this runcmd file.

```
icc_shell> set_host_options -target StarRC -submit_command runcmd
```

The StarRC tool can use distributed processing to perform extraction for multicorner-multimode designs. If you have a multicorner-multimode design and want to use StarRC distributed processing, use the `set_host_options -target StarRC` command to define the configuration. For details about using the `set_host_options` command to define a distributed processing configuration, see “[Configuring Distributed Processing](#)” on [page 2-14](#).

To verify the host option settings used for the StarRC tool, use the `report_host_options -target StarRC` command.

Automatically Generating the PrimeTime and StarRC Settings

The IC Compiler tool can automatically generate a script file that contains the required PrimeTime and StarRC settings for In-Design signoff timing analysis based on your current environment. To generate this script file, run the `create_signoff_setup` command.

Note:

For a successful StarRC or PrimeTime run, you still need to manually specify some of the settings, such as the location of a map file for the StarRC tool or a common setup file for the PrimeTime tool.

The `create_signoff_setup` command has the following syntax:

```
create_signoff_setup  
[-output directory] [-verbose] [-overwrite]
```

This command creates a file named `signoff_setup.tcl` file, which contains the required signoff settings. By default, the command saves this file in a directory named `signoff_setup`. To save the file to another directory, use the `-output` option to specify the directory. [Example 14-7](#) shows an example of the output Tcl file.

Example 14-7 Example signoff_setup.tcl File

```

set_primestime_options -exec_dir ... -common_file ...
set_starrcxt_options -exec_dir ... -map_file ...
report_primestime_options
report_starrcxt_options

# scenario-specific setup
remove_scenario xxx
create_scenario xxx
current_scenario xxx
set_primestime_options -sdc_file ... -specific_file ...
set_primestime_options -max_image ...
set_starrcxt_options -max_nxtgrd_file ...
report_primestime_options
...
run_signoff -check_only
check_signoff_correlation

```

The following list describes the commands used in the Tcl file.

- `set_primestime_options -exec_dir ${ptdir} -common_file ${common_file}`
`set_starrcxt_options -exec_dir ${starrcxtdir} -map_file ${map_file}`

The tool can detect a PrimeTime or StarRC binary from your search path. For a map file or common setup file, you need to specify the location manually.

- `set_primestime_options -sdc_file ${scenario_name}.sdc \`
`-specific_file ${scenario_name}.tcl`
`set_starrcxt_options -max_nxtgrd_file ${nxtgrd_max}`

These commands specify the necessary StarRC and PrimeTime settings. Some options require your manual input, such as the names of the nxtgrd and SDC files. Because the `create_signoff_setup` command writes out the SDC files during signoff using the settings in the IC Compiler or PrimeTime tools, you can decide whether to use an SDC file from the PrimeTime tool with the `-sdc_file` option, or let the tool automatically use the SDC file from the IC Compiler tool.

- `report_primestime_options`
`report_starrcxt_options`

These commands report the settings of the `set_starrcxt_options`, `set_primestime_options`, and `set_host_options` commands. You should always perform a report command for each scenario.

- `run_signoff -check_only`

Run a quick signoff without invoking the PrimeTime and StarRC tools.

For PrimeTime Users

If you are using the PrimeTime tool, download the Tcl script named signoff_setup.tcl.e.PT from [SolvNet article 031628, “Generating Required Settings for Sign-Off from PrimeTime Distributed Multi-Scenario Analysis,”](#) and run the script in the PrimeTime tool to generate a similar setup script.

To create the required settings for signoff within the PrimeTime tool, run the `create_signoff_setup` command using the following syntax:

```
source signoff_setup.tcl.e.PT
create_signoff_setup
  [-output directory] [-verbose]
  [-overwrite]
```

Note the following differences in this script as compared with the one generated in IC Compiler:

- For multicorner-multimode designs, if the IC Compiler scenarios differ from those in the PrimeTime tool, the script generated from the PrimeTime tool always removes all the IC Compiler scenarios and regenerates each scenario based on the PrimeTime environment.
- PrimeTime saved sessions are generated from PrimeTime distributed multi-scenario analysis. You can specify the saved sessions by using the `set_primateime_options` command.

Running the `run_signoff` Command

By default, the `run_signoff` command

- Requires the use of both the StarRC and PrimeTime signoff tools
 - To use the StarRC tool for extraction and the IC Compiler tool for timing analysis, use the `-star_only` option when you run the `run_signoff` command.
 - To use the IC Compiler tool for extraction and the PrimeTime tool for timing analysis, use the `-pt_only` option when you run the `run_signoff` command.

Note:

When you use the `-pt_only` option, the flow supports distributed multi-scenario analysis, AOCV modeling, and path-based analysis, as in the default `run_signoff` flow.

- Performs full extraction before starting the signoff flow

To quickly enter signoff mode without doing full extraction, use the `-skip_initial_extraction` option when you run the `run_signoff` command. To use this option, you must have a PrimeTime saved session that meets certain requirements.

For details, see “[Using A PrimeTime Saved Session To Skip Initial Extraction](#)” on [page 14-22](#).

Note:

The `-skip_initial_extraction` option is available only for the initial signoff run.

- Does not use the AOCV modeling feature in the PrimeTime tool

To enable the AOCV modeling feature in PrimeTime, use the `-aocvm` option when you run the `run_signoff` command. You must use a saved session or the `-specific_file` option when running In-Design signoff timing analysis with AOCV modeling. For more information about these settings, see “[Setting Up the PrimeTime Tool](#)” on [page 14-10](#).

- Does not use the PrimeTime path-based analysis feature

To improve timing analysis, use the PrimeTime path-based analysis feature. To enable this feature, use the `-path_based_analysis` option when you run the `run_signoff` command.

After you run the `run_signoff` command,

- The PrimeTime timing engine, rather than the IC Compiler timing engine, is enabled (unless you use the `-star_only` option).
- You cannot change the timing constraints.

To change timing constraints, exit signoff analysis mode by using the `run_signoff -signoff_analysis false` command, as shown in the following example:

```
icc_shell> run_signoff
icc_shell> report_timing
icc_shell> run_signoff -signoff_analysis false
icc_shell> set_false_path
...

```

- For multicorner-multimode designs,
 - You cannot activate a scenario that was inactive before you ran the `run_signoff` command. If you try to do so, the IC Compiler tool issues a PSYN-432 warning message.
 - You cannot use the `create_scenario` and `remove_scenario` commands. To enable these commands, exit signoff analysis mode by using the `run_signoff -signoff_analysis false` command.

Using A PrimeTime Saved Session To Skip Initial Extraction

To use the `-skip_initial_extraction` option with the `run_signoff` command, you must have a PrimeTime saved session with parasitic data for all conditions or scenarios.

For example, you must provide both maximum and minimum PrimeTime saved sessions for best-case and worse-case conditions, or one PrimeTime saved session for each scenario in a multicorner-multimode design. As with the signoff multicorner-multimode behavior, you specify the PrimeTime saved session by using the `set_primestime_options` command with the `-max_image` option in all scenarios, as shown in the following syntax:

```
set_primestime_options -max_image PT_session
```

In addition, you must set the `read_parasitics_load_locations` variable to `true` in the PrimeTime environment by using the following syntax:

```
pt_shell> set read_parasitics_load_locations true
```

If you are using the StarRC tool to generate the parasitic data for PrimeTime saved sessions, use the following StarRC settings.

For the SPEF format, use

```
REDUCTION: TOPOLOGICAL  
NETLIST_NODE_SECTION: YES  
EXTRA_GEOMETRY_INFO: NODE  
NETLIST_TAIL_COMMENTS: YES
```

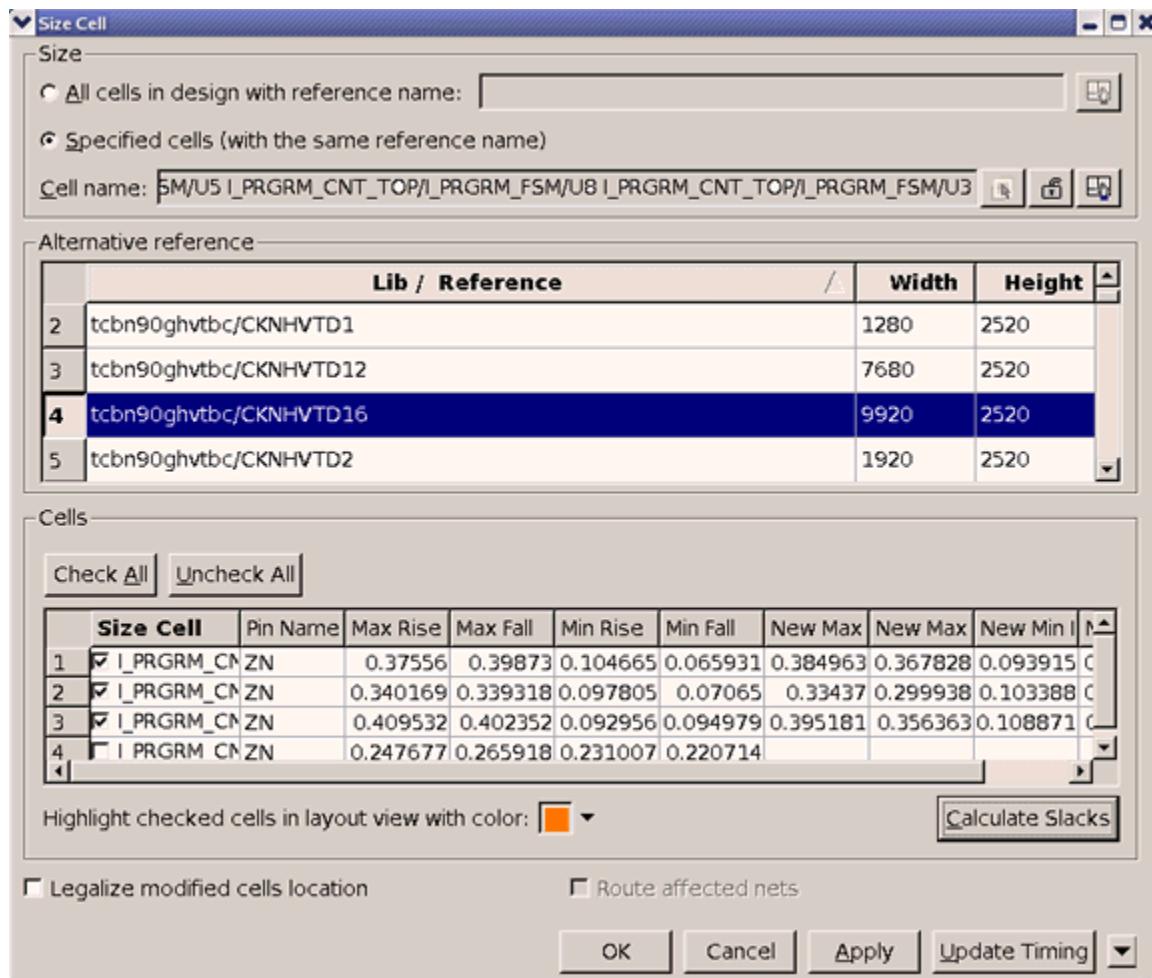
For the SBPF format, use

```
REDUCTION: TOPOLOGICAL  
EXTRA_GEOMETRY_INFO: NODE  
NETLIST_TAIL_COMMENTS: YES
```

Previewing Slack Improvements

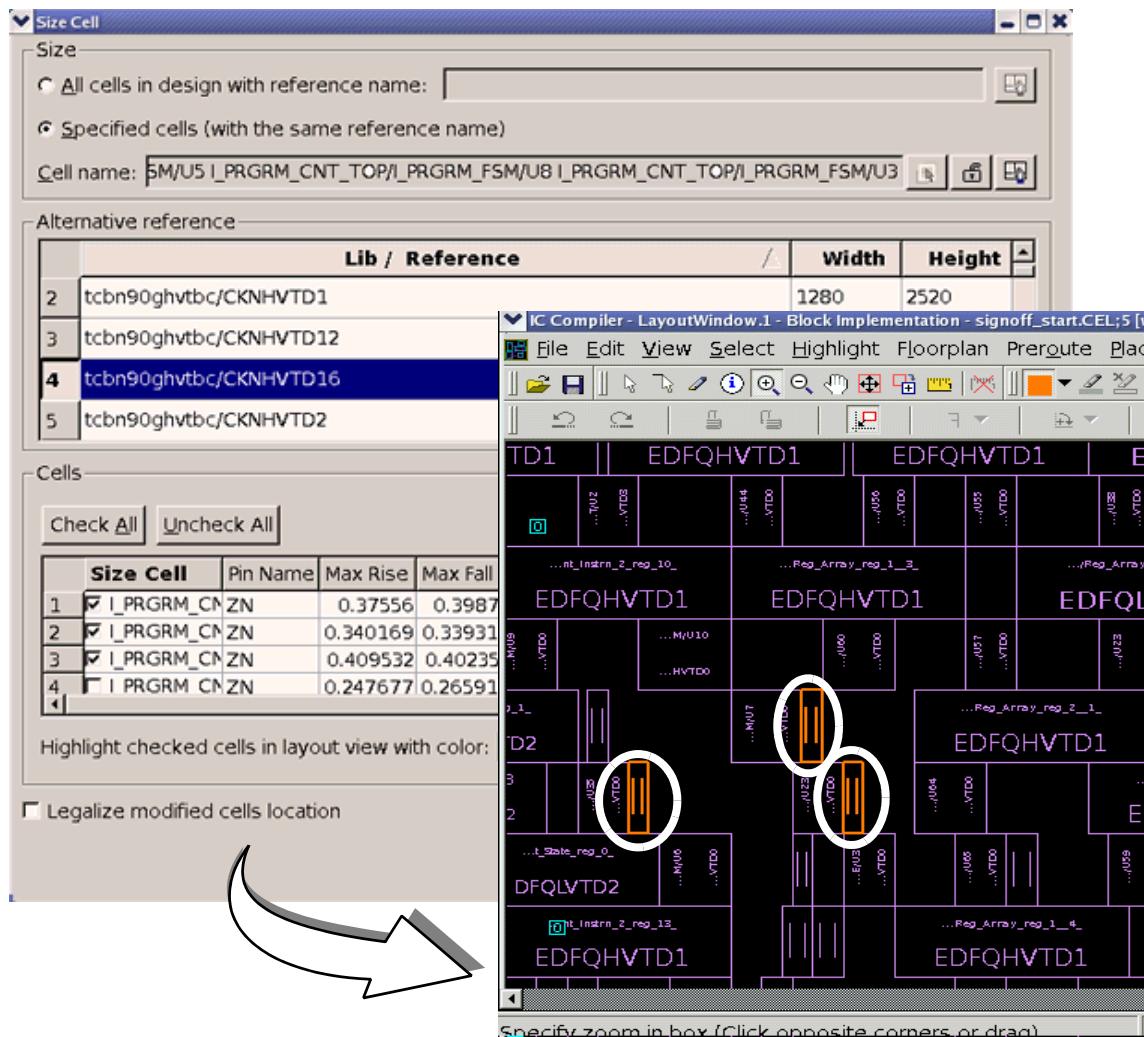
You can preview slack improvements in the GUI by using the Size Cell dialog box. Choose ECO > Size Cell. The Size Cell dialog box appears. Choose an alternative reference cell to size the cells in the design that have the same reference name. From this dialog box you can calculate slack values for alternative reference cells. This enables you to see how much slack can be improved with the new reference cell before you make actual changes to the netlist. [Figure 14-1](#) shows the current slack values and the new values for an alternative reference cell.

Figure 14-1 Slack Preview Using the GUI



From the Size Cell dialog box, you can highlight the checked cells in layout view to see the locations of these checked cells. You can select different colors for highlighting. [Figure 14-2](#) highlights the checked cells in yellow.

Figure 14-2 Highlighting Cell Locations



Troubleshooting and Problem Solving

Troubleshooting guidelines are as follows:

1. To help you find problems early and quickly, you can check your design by running the `run_signoff -check_only` command. In this case, you do not actually invoke the StarRC and PrimeTime tools.
2. If you are getting warnings about not being able to load the PrimeTime or StarRC tools, you might not be using the correct StarRC or PrimeTime versions or platforms.
3. In-Design signoff timing analysis requires that IC Compiler and PrimeTime tools use the same library models. If different library models, such as NLDM and CCS, are detected in the two tools, the `run_signoff` command issues an error message and quits. To ignore this inconsistency and allow these commands to continue, use the `-ignore_library_mismatch` option.
You can use the `get_libs *` or `write_link_library` command before the `run_signoff` command to ensure library consistency between the IC Compiler and PrimeTime tools.
4. After you save the Milkyway CEL view and reload it, you might have inconsistencies among the IC Compiler, PrimeTime, and StarRC timing reports. If this happens, run the `run_signoff` command again. The timing in the IC Compiler tool is different from the timing in the PrimeTime and StarRC tools, shown by the previous `run_signoff` session, until you run the `run_signoff` command again.

Checking the Signoff Correlation

The IC Compiler tool runs the PrimeTime and StarRC tools during In-Design signoff timing analysis. Run the `check_signoff_correlation` command to check the correlation between the IC Compiler and PrimeTime tools, and between the IC Compiler and StarRC tools, including

- The timing and signal integrity related variables, commands, and SDC setup in both the IC Compiler and PrimeTime tools

Note:

To check only the correlation between the IC Compiler and PrimeTime settings, use the `check_primestime_icc_consistency_settings` command instead of the `check_signoff_correlation` command.

- The extraction settings in both the IC Compiler and StarRC tools

When correlation checking is completed, the command generates a correlation report and other files for correlation analysis and correlation improvement.

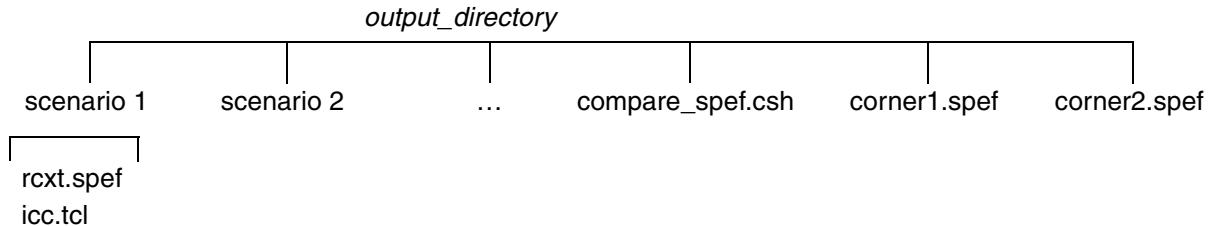
Note:

Correlation setup checking is faster with the `check_signoff_correlation` command than with the `run_signoff -correlation` command, because the `check_signoff_correlation` command does not invoke the StarRC or PrimeTime tool for setup checking.

Run the `check_signoff_correlation` command using the following syntax:

```
check_signoff_correlation  
[-setup] [-spef_out]  
[-star_only]  
[-preroute] [-keep_signoff_mode]  
[-html] [-output output_directory]
```

If the design being analyzed is a multicorner-multimode design, running the `run_signoff -correlation` or `check_signoff_correlation` command generates all the related reports and script files for each scenario. The generated files are stored in the output directory using the following structure:



A

Using GUI Tools

The IC Compiler GUI provides a variety of tools for viewing, analyzing, and editing your design. This appendix provides an overview of these tools. For detailed information about these tools, see IC Compiler Help.

This appendix contains the following sections:

- [Working With the GUI](#)
- [Viewing a Design Interactively](#)
- [Examining Design and Object Information](#)
- [Viewing the Physical Design](#)
- [Analyzing the Physical Design](#)
- [Analyzing Timing Paths](#)
- [Examining Routing and Verification Errors](#)
- [Editing the Physical Layout](#)

Working With the GUI

The IC Compiler GUI provides a flexible working environment with multiple windows for performing different tasks. The GUI windows operate independently in your X Windows environment, but they all share the design and global selection data in the tool.

You can open or close the GUI at any time during the session. For example, if you need to conserve system resources, you can close the GUI and continue the session in `icc_shell`. When you are ready to use the GUI again, you can reopen it.

For information about how to get started working with the IC Compiler GUI, see the following sections:

- [Opening the GUI](#)
- [Using GUI Windows](#)
- [Working in the Main Window](#)
- [Choosing Menu Commands in GUI Windows](#)
- [Working With the Console](#)
- [Viewing the Physical Layout](#)
- [Setting GUI Preferences](#)
- [Recording and Replaying a GUI Session](#)
- [Closing the GUI](#)
- [Exiting the IC Compiler Tool From the GUI](#)

Opening the GUI

Before opening the GUI, make sure your `$DISPLAY` environment variable is set to the name of your UNIX or Linux system display.

To open the GUI when you start `icc_shell`, use the `-gui` option when you start the tool. You can use this option with any of the IC Compiler packages listed in [Table 2-1 on page 2-2](#).

```
% icc_shell -gui
```

If you start `icc_shell` without the GUI, you can open the GUI by entering the `gui_start` command at the `icc_shell>` prompt.

To open the GUI, use the `gui_start` command.

```
icc_shell> gui_start
```

When you open the GUI, it automatically runs the commands in the GUI setup files named .synopsys_icc_gui.tcl. For information about IC Compiler setup files, see [“Starting the IC Compiler Tool” on page 2-2](#).

Note:

In addition to running the setup files when you open the GUI, the tool loads application preferences from a file named .synopsys_icc_prefs.tcl. For details, see [“Setting GUI Preferences” on page A-22](#).

You can also specify a script to run by using the -f option when you run the `icc_shell` command or the `-file` option when you run the `gui_start` command. For example,

```
icc_shell> gui_start -file my_gui_script.tcl
```

If you open a design in `icc_shell` and then open the GUI, the tool automatically opens a layout window and displays the design in the layout view. This is the primary design in the window. If you have multiple designs open when you open the GUI, the tool opens a separate layout window for each design. The active window (the window that has the mouse focus) is the window with the current design.

When you open the GUI, it builds query data by default if a design cell is already open. This query data ensures consistent, fast interactive query operations. If you open the GUI before opening a design cell, the GUI builds the query data the first time you open a design.

If you prefer not to wait for the tool to build the query data when you open the GUI or open a design in the GUI, you can disable the build by setting the following variable:

```
icc_shell> set_app_var gui_build_query_data_table 0
```

You can build the query data manually at any time by choosing **Edit > Build Query Data** in a layout window.

See Also

- [Starting the IC Compiler Tool](#)

Using GUI Windows

Each top-level GUI window displays design information in view windows and provides menus and dialog boxes (with keyboard shortcuts), toolbars and panels, interactive mouse button tools, a command console (with a command-line interface), and a status bar. View windows are child windows that appear in the workspace area between the toolbars and the status bar.

The IC Compiler GUI provides the following types of windows:

- IC Compiler main window
- Layout window
- Timing analysis window
- Interactive clock tree synthesis (CTS) window
- Relative placement (RP) window
- Partition editor window

Each top-level GUI window is independent of other GUI windows. The title bar displays the product name (IC Compiler), the window name (MainWindow.1 for the window that appears when you start the GUI) and the name of the active view (the view window that has the mouse focus). Design objects you select in one window are automatically selected in the other windows.

You can open and close GUI windows, and you can open multiple instances of each window to work with different design settings. Each window instance has a unique name that includes an incremented number.

To open a new instance of a GUI window,

- Choose Window > New *window-type*.

To close a GUI window,

- In the window you want to close, choose Window > Close Window.

Note:

If you close all the open GUI windows, you end the GUI session but you remain in the `icc_shell` and your designs remain open. You can reopen the GUI by using the `gui_start` command.

You can move, resize, minimize, or maximize GUI windows by using the window management tools on your UNIX or Linux desktop.

For more information about the top-level GUI windows, see the following sections:

- [Menu Bar](#)
- [Toolbars](#)
- [Status Bar](#)
- [View Windows](#)
- [Panels](#)

Menu Bar

The menu bar at the top of a GUI window contains menus with the commands you need to work in the window. You can display a brief message in the status bar about the action that a menu command performs by holding the pointer over the command. For menu commands that can also be used by clicking a toolbar button or pressing a keyboard shortcut, the menus show representations of those alternatives.

Note:

If a window is not wide enough to display all the menu names on the menu bar, the window displays all the menu names that fit, from left to right, followed by an overflow button (»). To access the other menus, click the overflow button.

Toolbars

Each top-level IC Compiler window provides toolbars with buttons you can use to quickly perform frequently used operations or tasks. To determine the function of a toolbar button, hold the pointer over the button. A ToolTip displays the name of the button, and the status bar displays a brief description of its use. You cannot disable these messages.

Toolbars are always attached to a window edge. If a window edge is not long enough to display all of the toolbars attached to it, the GUI displays the full toolbars that fit and shortened versions of the other toolbars. A shortened toolbar displays an overflow button (»). To access the other buttons on a shortened toolbar, click the overflow button.

You can move a toolbar to another location by dragging the double bars on one of its sides. You can also disable a toolbar, hiding it from view.

To display or hide a toolbar or panel,

- Choose View > Toolbars > *toolbar_name*.

A check mark beside the name of a toolbar on the Toolbars menu indicates that the toolbar is visible.

Status Bar

Each GUI window displays a status bar at the bottom of the window. The status bar displays the following information:

- Status information such as the number and type of selected objects
- Information about the action performed by the toolbar button, menu command, or workspace tab under the pointer
 - If you hold the pointer over a menu command, a toolbar button, or a tab in the workspace, the status bar displays a brief message about the action that the command, button, or tab performs.
- The relative coordinates of the pointer position over an active layout view (layout windows only)

To quickly display the list of selected objects in the Selection List dialog box, click the  button at the right end of the status bar.

You can control the visibility of the status bar in each GUI window.

To display or hide the status bar,

- Choose View > Status Bar.

A check mark next to the Status Bar command on the View menu indicates that the status bar is visible.

View Windows

The IC Compiler tool displays design information in view windows, which are child windows that open within the workspace area of a GUI window. View windows display specific graphic or textual design information.

Most menu commands and toolbar buttons in a GUI window operate in the active view, which is the view window that has the mouse focus. Some view windows contain two or more panes, with a view in each pane. You can drag the split bars between two panes to adjust their relative sizes within the window.

You can resize a view window or move it to another location inside the workspace area of its parent window.

To change the size of a view window,

1. Hold the pointer over an edge or corner of the window until the pointer changes shape.
2. Drag the edge or corner to resize the window.

To move a view window,

1. Move the pointer over the title bar.
2. Drag the window to a different location.

You can organize view windows by tiling or cascading them within the workspace area.

To tile the view windows in the workspace area,

- Choose Window > Tile Views.

To cascade the view windows in the workspace area,

- Choose Window > Cascade Views.

You can also minimize individual view windows, or maximize a view window to fill the workspace area.

The GUI displays a tab at the bottom of the workspace area for each open view window. When you click a tab, the view window appears on top of the other view windows in the workspace area and becomes the active view.

Note:

If a view window and a panel overlap on the screen, the panel appears on top of the view window.

In most views, you can use navigation keys (the arrow keys, Page Up, Page Down, Home, and End) to scroll through and navigate the view.

Panels

Panels are enhanced toolbars that open within the workspace area of a top-level window to provide tools or views for performing particular tasks. Most panels are associated with a particular view window and operate on the active view (the view window that has the mouse focus). An exception is the console, which contains a command line and its own views.

Some panels contain tabs that you can click to access different tools or views. The first time that you open a panel during a session, it displays the tools or view for its default tab.

You can move a panel to another location on or off the parent window by dragging the double bars on one of its sides. You can also dock a panel (attach it to a window edge) or let it float where you leave it.

You can also disable a panel, hiding it from view. (A panel that is associated with a particular view is automatically hidden when you close the view window.)

To display or hide a toolbar or panel,

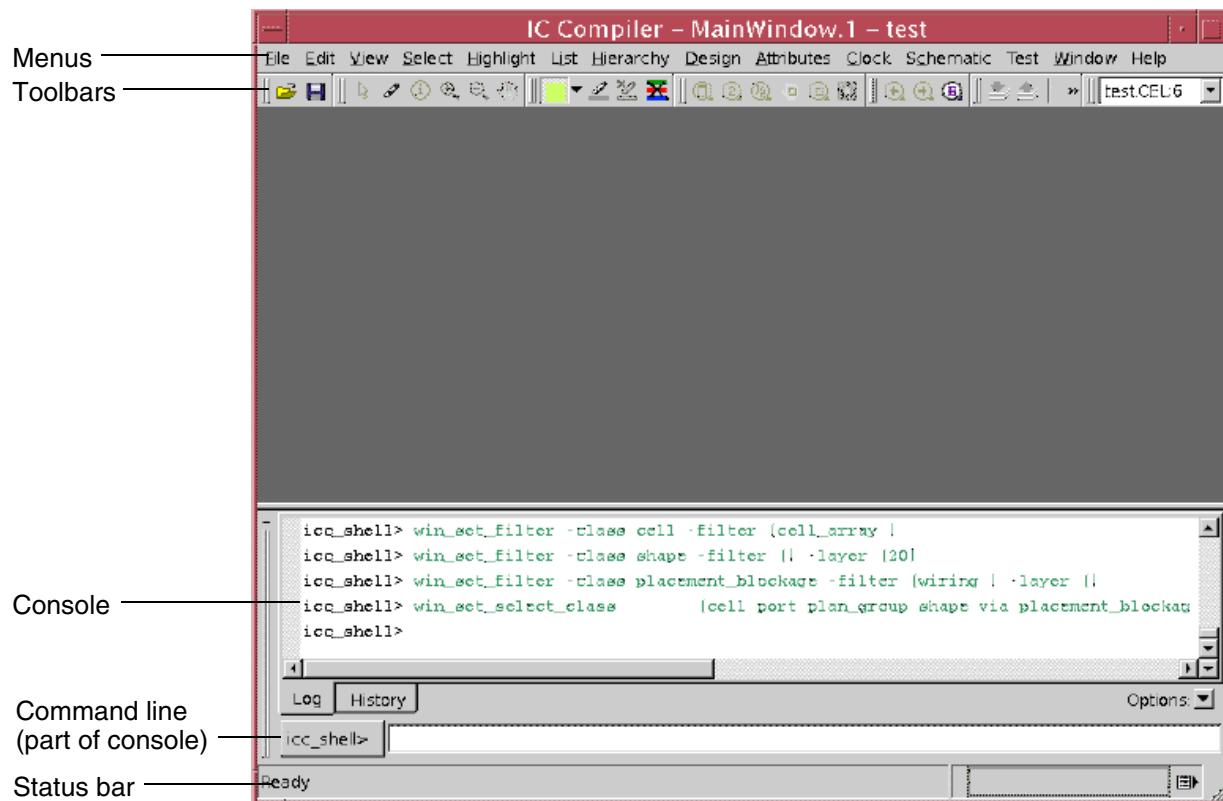
- Choose View > Toolbar > *panel_name*.

A check mark beside the name of a panel on the Toolbars menu indicates that the panel is visible.

Working in the Main Window

When you start the IC Compiler tool and open the GUI, the IC Compiler main window appears. [Figure A-1](#) identifies the major features of the main window. For information about these features, see the “IC Compiler Main Window” topic in IC Compiler Help.

Figure A-1 IC Compiler Main Window Features



In the main window, the console is docked above the status bar by default. Although the other top-level GUI windows also provide consoles, they are hidden by default.

Typically, you use the main window to

- Run Tcl scripts
- Enter `icc_shell` commands, monitor command processing, and view messages in the console
- Read in and save (or close) designs
- Set variables and setup options
- Open other GUI windows
- Close the GUI or exit the tool

In addition, you can open view windows (in the workspace area below the toolbars), explore the logic design in the hierarchy browser, examine the logic design in design schematics, and examine design object and timing information in object list views.

Choosing Menu Commands in GUI Windows

The IC Compiler GUI provides menu commands and dialog boxes for most graphic features, such as viewing, selecting, and highlighting objects in layout views. In addition, the GUI provides menu and dialog box equivalents for many `icc_shell` commands. Menu commands are grouped by function on the menus in each GUI window. You can add commands to these menus and even create new menus.

To choose a command on a menu bar menu,

- Click the menu name to open the menu, and click the command name on the menu.

Some frequently used menu commands are also available on pop-up menus for individual views.

To choose a command on a pop-up menu,

- Move the pointer over the object of interest, right-click to display the menu, and click the command name.

A menu command can perform an immediate operation, display a submenu, or display a dialog box.

- Menu commands that display a submenu are followed by a right-pointing arrow.
- Menu commands that open a dialog box that requires a response before performing an operation are followed by an ellipsis (...). These commands open a dialog box to prompt you for the information.

Dialog boxes that require a response before performing an operation contain OK and Cancel buttons and sometimes an Apply button. After selecting options or entering information in the dialog box, you respond by clicking OK or Apply.

- Menu commands without an arrow or ellipsis either perform an immediate operation or open a dialog box that performs immediate operations.

Dialog boxes that perform immediate operations usually display options and contain a Close button. You select options that perform operations and click Close to close the dialog box.

When you choose a command or select a dialog box option that performs an immediate action, or when you click OK or Apply in a dialog box that requires a response, the tool displays command output (including processing messages and any warnings or error messages) in both `icc_shell` and the console log view.

This manual identifies commands with their menus in the following formats:

- *Menu > Command*
- *Menu > Submenu > Command*

where

- *Menu* represents a menu title on the menu bar.
- *Submenu* represents a menu command that displays a submenu. Some submenus contain commands that open other submenus.
- *Command* represents a command that performs an operation or displays a dialog box.

Each menu command can also be activated by a shortcut key, which is indicated on the menu by an underscore (_) below a letter in the command and, if needed, the name of the modifier key (Shift or Ctrl) to the right of the command name. You can view a list of shortcut keys by choosing Help > Report Hotkey Bindings. For details, see “[Displaying the List of Keyboard Shortcuts](#)” on page A-12.

Finding Menu Commands and Dialog Boxes

You can quickly find the equivalent menu command and open the dialog box for any `icc_shell` command by using the `gui_show_form` command. You can specify a single `icc_shell` command or use wildcard characters (?) or (*) to specify commands with similar names. The `gui_show_form` command has the following syntax:

```
gui_show_form command_name_or_name_pattern
```

When you specify a command that is used by only one dialog box, the GUI displays the menu command and the application task in the console log view and opens the dialog box.

```
icc_shell> gui_show_form open_mw_cel
Info: GUI access for open_mw_cel
      Menu : File->Open Design...
      Task : all
```

Task indicates the application tasks in which the menu command is available. For menu commands in the layout window, you can set the current application task to block implementation, design planning, or all tasks. For menu commands in other windows, the current application task is always all tasks. To learn more about application tasks in the layout window, see “[Setting the Current Task](#)” on page A-21.

When you specify an icc_shell command that is used by more than one dialog box or specify a name pattern that matches more than one icc_shell command name, the GUI displays an information table in the man page viewer. For each icc_shell command, the table displays information in the following columns:

- Command – displays the icc_shell command names with keywords for additional menu commands
- Menu – displays the equivalent menu command names
- Window – displays the names of the windows in which the menu commands can be found
- Task – displays the application tasks in which the menu commands are available

You can click links in the man page viewer to open dialog boxes and view man pages.

To open the dialog box for a menu command,

- Click the link on the menu command name in the man page viewer.

To display the man page for an icc_shell command,

- Click the link on the icc_shell command name in the man page viewer.

If you specify an icc_shell command used by a menu command that performs an immediate action without opening a dialog box, the GUI displays the command information in the man page viewer. If you enter a name for an icc_shell command that does not exist, the GUI displays an error message in the console log view.

Previewing `icc_shell` Commands in Dialog Boxes

Many IC Compiler dialog boxes support a preview mechanism that you can use to view the commands without running them. You can view the command equivalents for various dialog box options, or combinations of options, or generate an `icc_shell` command that you want to copy into a script. When you enable the preview mechanism, it remains on for all dialog boxes that support the feature until you disable it.

When you click OK or Apply in a dialog box that supports the preview mechanism while the mechanism is enabled, the `icc_shell` command equivalents appear in the console history log preceded by a pound sign (#), but the tool does not execute the commands and they do not appear in the console log view or the shell transcript.

To enable or disable the dialog box command preview mechanism,

- Click the ▾ menu button in the bottom right corner of a dialog box, and choose Preview command only.

A check mark next to “Preview command only” on the menu indicates that the preview mechanism is enabled.

Displaying the List of Keyboard Shortcuts

You can view a report of the keyboard shortcuts for Tcl commands and commands on menus in the active window. The tool displays the hotkeys report in a report view.

To display the report of keyboard shortcuts for the active window,

- Choose Help > Report Hotkey Bindings.

The report consists of three columns:

- Hot Key lists the shortcut keys or key combinations.
- Type indicates whether the key is a shortcut for a menu command or a Tcl command.
- Function lists the commands that the shortcut keys launch.

Working With the Console

The console provides a command-line interface and two views, a log view that displays the session transcript (the default) and a history view that displays the command list.

You can use the console to

- Enter IC Compiler Tcl commands on the console command line
- View either the session transcript (log view) or the command history list (history view) by clicking the tabs above the command line

- Display an error message man page in the man page viewer by clicking the message number in the console log view
- Copy and edit or reuse commands
- Search for, select, and save commands or messages in the log view or the history view

You can enter any IC Compiler shell command on the console command line. For example, if you enter `get_selection`, the console log view displays a list of the names of all selected objects. When you enter a command on the console command line, make sure the console has the mouse focus.

To enter a command on the console command line,

1. Type the command.
2. Click the `icc_shell>` prompt button or press Return.

You can open (or close) one console in each top-level window. When the console is open, you can dock it to the bottom or top of its parent window, or move it over or away from the window.

For more information about the console, see the following sections:

- [Viewing the Session Log](#)
- [Viewing the History View](#)
- [Previewing `icc_shell` Commands in Dialog Boxes](#)

Viewing the Session Log

The console log view displays a transcript of session information that includes the commands you entered and the IC Compiler output and messages resulting from your commands. You can use this information to check on tool status after performing functions, to troubleshoot problems you encounter, and to look up information about past functions. You can even reenter commands you have already used by copying them from the log to the `icc_shell` command line.

To view the session transcript,

- Click the Log tab in the console.

The console changes to the log view, which is displayed by default whenever the console is opened.

The log view displays the commands you enter next to a boldfaced `icc_shell>` prompt. Warnings and error messages are noted with “Warning” or “Error” as the first word. If you need to see information not currently displayed in the display area of the log view, use the scroll bars to scroll through the session information.

You can use commands on the Options menu (on the right side of the console above the command line) to

- Find text in the transcript
- Select and copy text in the transcript
- Search the transcript for commands or messages
- Save text from the transcript in a text file

To find text in the transcript,

1. Choose Find from the Options Menu or Context Menu.

The Find Text in Console dialog box appears.

2. Type the text you want to find in the Find box.

You must type the text exactly as it appears in the transcript, including uppercase and lowercase letters and blank spaces.

3. Search forward in the transcript by clicking Next, or search backward by clicking Previous.

The console highlights the next or previous instance of the text. Repeat this step as needed to find additional instances of the text.

To copy a used command from the log view to the command line,

1. Select the command or portion of the command that you want to copy in the log view.
2. Middle-click the console command line.

If you already have text on the command line, middle-click where you want to paste the selected information.

You can copy text in the transcript that you want to paste somewhere else, such as in a file.

To copy text in the transcript,

1. Drag the pointer over the text to select it.

You can select all of the text in the transcript by choosing Options > Select All.

2. Choose Copy from the Options Menu or Context Menu.

You can search the transcript for commands, error messages, warning messages, or informational messages. To search the transcript,

1. (Optional) Set up the search criteria to include or exclude certain types of information by choosing commands on the Options menu.

You can include or exclude commands, error messages, warnings, or information messages. Check marks appear on the Options menu next to the commands for the types of text included in the search.

2. Search forward or backward by choosing Find Next or Find Previous from the Options or Context menu.

The console highlights the next or previous instance of any text type you included in the search.

3. Repeat step 2 as needed to find the command or message you need.

You can save all the text in the transcript, save selected text, or save just the error and warning messages.

To save text from the transcript in a text file,

1. Do one of the following from the Options or Context menu:

- To save the transcript, choose Save Contents As.
- To save selected text, drag the pointer over the text to select it, and choose Save Selection As.
- To save error and warning messages, choose Save Errors/Warnings As.

2. In the dialog box that appears, navigate to the directory where you want to save the file.
3. Enter the file name in the “File name” box.
4. Click Save.

Viewing the History View

The console history view lists shell, menu, and stroke commands you have used in the current session. You can do the following with this list:

- See which commands you have used
- Find and reuse commands already used
- Copy commands in the list
- Save the list for later use

To view the command history for the current session,

- Click the History tab in the command console.

The console displays the list of commands, each with a number showing the order in which you used it.

If you need to see information not currently displayed in the display area of the history view, use the scroll bars to scroll through the list of commands. Alternatively, you can enter the `history` command on the command line. This displays the list of commands in the log view.

To reuse a command listed in the history view, you can do any of the following:

- Double-click the command in the history view.
- Select the command in the view and click either Execute (to immediately rerun the command) or Edit (to paste the command on the command line where you can edit it).
- Use the `history` command to list the commands you entered during this session. To recall a specific command from the history list, enter an exclamation point (!) followed by the number of the command on the `icc_shell` command line. For example, to reuse command number 15 in the list, enter the following:

```
icc_shell> !15
```

You can copy commands in the list that you want to paste somewhere else, such as in a file.

To copy commands in the history list,

1. Drag the pointer over the text to select it.

You can select all of the text in the transcript by choosing Options > Select All.

2. Choose Copy from the Options or Context menu.

To save the command history list (or a portion of the list) in a text file,

1. (Optional) To save a portion of the list, select the commands you want to save.

2. Do one of the following:

- If you are saving the entire list, click the Save Contents As button or choose Options > Save Contents As.

The Save Contents As dialog box appears.

- If you are saving just a portion of the list, choose Options > Save Selection As.

The Save Selection As dialog box appears.

3. If necessary, navigate to the directory where you want to save the file.

4. Enter the file name in the “File name” box.
5. Click Save.

You can set options in the Application Preferences dialog box to control which types of GUI commands are included in the history list. For details, see [“Setting GUI Preferences” on page A-22](#).

Using Online Help

The IC Compiler GUI provides the following types of online Help:

- A man page viewer that allows you to view, search, and print man pages for commands, variables, and error messages
- Contextual Help pages that display information about the current interactive editing or visualization tool in GUI
- A Web browser-based Help system that explains how to use the tool with the GUI

Using the Man Page Viewer

You can use the man page viewer to view, search, and print man pages for commands, variables, and error messages.

To view a man page in the man page viewer,

1. Choose Help > Man Pages.

The man page viewer appears and displays a list of links for the different man page categories.

2. Click the category link for the type of man page you want to view: Commands, Variables, or Messages.

The contents page for the category displays a list of title links for the man pages in that category.

3. Click the title link for the man page you want to view.

You can also display man pages in the man page viewer by using the `man` or `gui_show_man_page` command on the console command line.

Note:

If you enter the `gui_show_man_page` command in `icc_shell` when the GUI is closed, the tool automatically opens the GUI and displays the man page in the man page viewer.

For more information about using the man page viewer, see the “Viewing Man Pages” topic in IC Compiler Help.

Using Contextual Help

Contextual Help pages are available for most of the interactive mouse tools and for certain other tools, such as the error browser. These Help pages provide usage information for the tool, explanations of the tool's options, and descriptions of the Tcl command options used to log the tool operations.

To view the Help page for the tool you are currently using,

- If the tool displays options on the Mouse Tool Options toolbar, click the  button.
- If the tool displays options on a panel or in a dialog box, click the Help button.

Using the Help System

IC Compiler Help is a browser-based HTML Help system that provides detailed information and instructions for using the IC Compiler GUI. You can use it to

- Learn about tool features, including windows, views, toolbars, panels, menus, and dialog boxes
- Learn how to use the interactive visualization and editing tools
- Learn how to use the IC Compiler tool to perform design tasks

To open IC Compiler Help in your Web browser,

1. Choose Help > IC Compiler Online Help.

The Help system appears in a Web browser window and displays the Welcome topic for IC Compiler Help.

2. Use the navigation frame (leftmost frame) to find the information you need in one of the following ways:

- Browse for the topic in the hierarchical table of contents by clicking Contents and expanding the appropriate books until you find the information you need.
- Find the topic by its subject by clicking Index and looking for the subject in the alphabetical listing.
- Search for text found in topics by clicking Search and entering the text.

If more than one topic has the words you are searching for, you must select the appropriate topic from a list of topics.

For more information, see the following sections:

- [Searching for Text](#)
- [Starting the Help Browser](#)

Searching for Text

IC Compiler Help lets you search for text in any Help topic. By default, the Search mechanism searches for one or more words you enter in the navigation frame. To search for an exact phrase containing two or more words, enclose the words within double quotation marks ("").

A legal search string has the following characteristics:

- Contains letters and numbers
- Can begin with a period (.) or hyphen (-)
- Can contain underscores (_), but only in a string that begins with a hyphen
However, you can find a string that contains underscores by replacing them with blank spaces and enclosing the string in double quotation marks.
- Can contain a slash (/), backslash (\), or colon (:), but only when preceded and followed by whole words or in a string that begins with a hyphen
- Cannot contain a comma (,) or semicolon (;) in any position
- Cannot contain quotation marks (" , ') or any other special characters in any position
- Is three or more characters long

Searches are not case-sensitive. Search results are ranked according to the location of the matched term and the number of matches. Matches in headings always rank near the top. To locate the search string in a Help topic, you can use the quick search box on the button bar above the topic frame or the Find command in your Web browser.

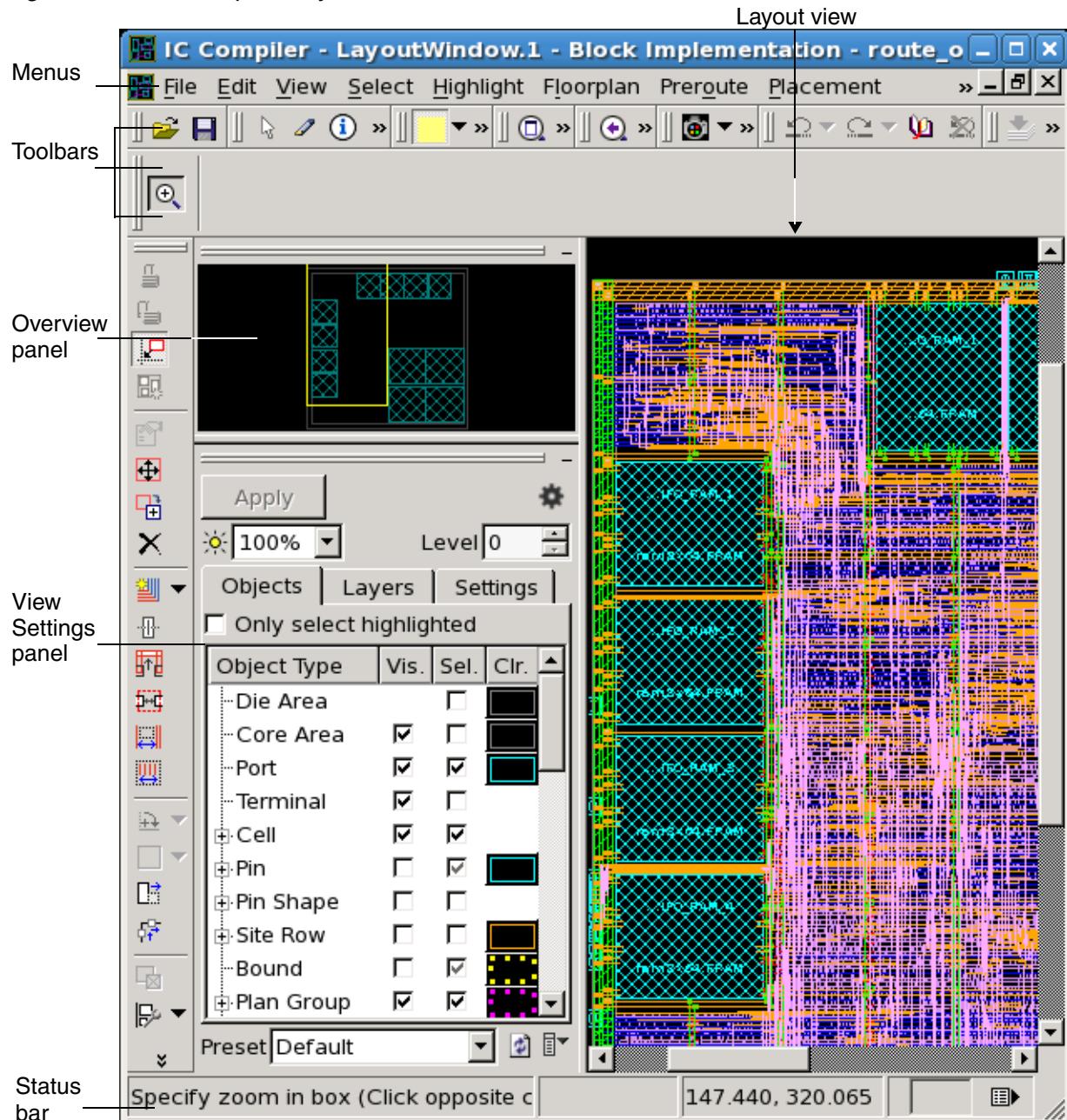
Starting the Help Browser

You can open IC Compiler Help from within the IC Compiler GUI (choose Help > IC Compiler Online Help) or standalone in your Web browser. To open the Help system from within the GUI, the browser executable file must be specified in your UNIX or Linux path variable.

IC Compiler Help makes extensive use of JavaScript and style sheets. If your browser encounters problems displaying the Help system, open the browser preferences and make sure that JavaScript and style sheets are enabled and that JavaScript is not blocked by your security preferences.

Viewing the Physical Layout

When you read in a design, a new layout window appears and displays the design in the layout view. This is the primary design in the window. Each layout view you open in a layout window displays the primary design. [Figure A-2](#) identifies the major layout window features.

Figure A-2 IC Compiler Layout Window Features

The layout window provides the physical design working environment for the GUI. Layout views provide the focal points for viewing, analyzing, and editing the physical design.

You can use the layout window for each major stage in the physical design process, including the floorplan, the power network, placement, preroute, routing, finishing, and verification.

The layout window provides the following layout analysis tools:

- Visually customizable layout views
- A hierarchy browser you can use to explore the design hierarchy and highlight hierarchical cells and subblocks in the layout view
- A relative placement hierarchy view you can use to examine relative placement groups and hierarchy levels in the design
- An error browser you can use to examine routing and verification errors and highlight their locations in the layout view

Setting the Current Task

You can configure the IC Compiler GUI for the design task you are performing by setting the current application task to block implementation, design planning, or all tasks. The block implementation and design planning tasks simplify the menus you use and provide better layout view rendering performance for large designs.

The current task applies to all open layout windows. It controls the types of menus and menu commands that appear in the layout window menu bar and the visibility of some object types in the layout view.

To set the current task,

- Choose File > Task > *task_type*, where *task_type* is Design Planning, Block Implementation, or All Tasks.

Alternatively, you can use the `gui_set_current_task` command. You can also change the default task by setting the current task in one of the IC Compiler setup files, `.synopsys_icc.tcl` or `.synopsys_icc_gui.tcl`, in the design directory or in your home directory.

When you set the current task to design planning, the tool automatically hides standard cells and cell contents in the layout view and displays the following message in the console log view:

Visibility is turned OFF for cells and cell contents because the task is set to Design Planning (GUI-026)

When you set the current task to either block implementation or all tasks, the tool automatically displays standard cells and cell contents in the layout view and displays an appropriate message in the console log view.

You can disable this automatic behavior by setting the `setVisibilityByTask` preference to false. For example, to set the preference, enter

```
icc_shell> gui_set_pref_value -category layout \
           -key setVisibilityByTask -value false
```

In addition, you can manually control the visibility of cells and cell contents by setting options on the View Settings panel. For details, see the “Setting the Current Task” topic in IC Compiler Help.

Setting the Primary Design

If you open a new layout window when multiple designs are open, you must select the primary design for the window.

To open a new layout window,

- Choose Window > New Layout Window.

If you have more than one design open, the Choose Design dialog box appears and displays a list of the open designs.

- a. Select the name of the design you want to make the primary design for layout views in the new layout window.
- b. Click OK.

Note:

The GUI provides special windows for timing path analysis and clock tree synthesis. For details, see [“Analyzing Timing Paths” on page A-85](#) and [“Analyzing Clock Trees” on page A-83](#).

Setting GUI Preferences

At the beginning of a GUI session, the tool loads GUI preferences and layout view settings from the preferences file, .synopsys_icc_prefs.tcl, in your home directory. You can change global preferences for GUI tools and the window environment by setting options in the Application Preferences dialog box. If you change preference settings during a GUI session, the new settings are automatically saved in your preferences file when you click OK in the Application Preferences dialog box.

The preferences you can set include

- Font options
- GUI command logging options
- Global tool default options
- Global layout view options
- Global schematic view options

For details about setting these preferences, see the “Setting GUI Preferences” topic in IC Compiler Help.

Recording and Replaying a GUI Session

You can use the command log file as a replay script to reproduce a GUI session in the tool. The command log file records the `icc_shell` commands processed by the tool, including setup file commands and variable assignments. Before you record a GUI session, make sure that the preferences for logging selection operations and interactive GUI operations are enabled in the Application Preferences dialog box.

Important:

You must replay the entire command log file as is, with no modifications, to reproduce the exact state of the design in the GUI.

To set GUI command logging preferences,

1. Choose View > Preferences.

The Application Preferences dialog box appears.

2. Select Global Settings in the Categories tree.

3. Select the optional output logging options as needed.

- To enable logging of interactive selection operations, select the “Selection commands” option.
- To enable logging of other interactive GUI operations, select the “GUI commands” option.

4. Click OK or Apply.

For more information about the command log file, see *Using Tcl With Synopsys Tools*.

Closing the GUI

You can open or close the GUI at any time during the session. For example, if you need to conserve system resources, you can close the GUI and continue the session in `icc_shell`.

To close the GUI, do one of the following:

- Choose File > Close GUI in the main window or layout window.
- Choose Window > Close All Windows in any GUI window.
- Enter the `gui_stop` command.

```
icc_shell> gui_stop
```

Exiting the IC Compiler Tool From the GUI

To exit the tool, enter the `exit` or `quit` command on the console command line or choose File > Exit in the main window or layout window. When you exit the tool by using the command line, a message box appears. Click OK to close the message box and exit the application.

When you exit the tool by choosing File > Exit and there are unsaved changes in any open designs, the Exit IC Compiler dialog box appears, which allows you to save the changes to the modified designs. By default, you can either save or discard all changes. If you select “Show advanced options,” you can

- Specify a list of scenarios with constraints that you need to save with the designs.
- Select from the following options for each modified design,
 - Save a new CEL version of the design
 - Save a copy of a design with a new name
 - Save the design hierarchy (a modified top-level design together with all its subdesigns and attached files) into the same CEL version
 - Close the design after saving or discarding changes

For more information about using the advanced options to save or discard changes, see [“Saving the Design in the GUI” on page 3-61](#).

If you do not have unsaved design changes, an IC Compiler message box appears instead of the Exit IC Compiler dialog box. If you see this message box, click OK to exit.

The IC Compiler tool does not save view settings when you exit. If you have changed view settings for layout or schematic views and want to save them, do so before exiting the tool, as explained in [“Setting and Saving View Properties” on page A-61](#).

Viewing a Design Interactively

The IC Compiler GUI displays graphic representations of design data in several types of views. To help you visualize a design, the GUI provides tools and commands that you can use to examine the design data and to select or highlight objects of interest for analysis and editing.

- A layout view displays graphic representations of design objects in a single, flat view of the physical design. You can display or hide objects or layers and customize the viewing environment. You can also open multiple layout views to work simultaneously with different areas of the design.

For information about working in a layout view, see “[Viewing the Physical Design](#)” on page [A-56](#).

- Histograms are focal points of visual timing analysis that allow you to view the overall timing performance of your design. You can generate histograms to view distributions of timing values such as slack or clock arrival times.

For information about working with histograms, see “[Viewing Histograms](#)” on page [A-87](#).

- Schematic views display schematic representations of specific types of design data. You can use different types of schematic views to examine logic designs (hierarchical cells), selected objects or timing paths (including fanin and fanout logic), and the fanin or fanout networks for selected clock trees.

For information about working in schematic views see “[Viewing Path and Design Schematics](#)” on page [A-89](#) and “[Examining Fanout or Fanin Paths in the Clock Tree Network](#)” on page [5-151](#).

- Abstract clock graph views display the relationships between the driving cells and output loads in a clock network or the relationships between clock trees. An abstract clock graph view displays a symbolic representation of design data that is similar to a schematic display. You can examine

- Clock skew, bottlenecks, insertion delays, and intrinsic path delays in a latency abstract clock graph view that facilitates clock tree latency and skew analysis
 - Clock tree fanout structures in a leveled abstract clock graph view

For information about working in abstract clock graph views, see “[Viewing Abstract Clock Tree Graphs](#)” on page [5-153](#).

The GUI provides interactive left-button mouse tools that you can use to view and probe a design. In a layout or schematic view, you can

- Select or deselect individual objects or timing paths or objects in an area of the design
Selection is global in IC Compiler. Any objects or timing paths that you select appear selected in all open views.
- Color highlight individual objects or timing paths or objects in an area of the design
Highlighting is global in graphic views. When you highlight objects or timing paths in a view, the GUI highlights them with the same color in every view where they appear.
- Display (query) object properties for individual objects (such as design and timing attribute values)
- Traverse (pan) the design and magnify or shrink (zoom) areas of interest
- Draw rulers to measure distances in a layout view or clock graph view

For information about performing these tasks, see the following sections:

- [Selecting a Mouse Tool](#)
- [Previewing Objects](#)
- [Selecting Objects](#)
- [Viewing Object Information](#)
- [Highlighting Objects](#)
- [Selecting Only the Highlighted Objects](#)
- [Selecting or Highlighting Objects by Name](#)
- [Magnifying and Traversing the Design](#)

For details about drawing rulers, see “[Drawing Rulers](#)” on page [A-58](#).

Selecting a Mouse Tool

Mouse tools control the actions performed in graphic views when you click or drag the pointer with the left mouse button. IC Compiler provides mouse tools for design visualization that you can use to

- Magnify or traverse the view in a layout, schematic, histogram, or clock graph view
- Select, highlight, or query objects in a layout, schematic, or clock graph view
- Draw rulers in a layout or clock graph view

The visualization mouse tools are available in all top-level GUI window. You can activate these mouse tools by clicking buttons on the Mouse Tools toolbar or by choosing commands on the Mouse Tools menu. (Choose **View > Mouse Tools** to display the menu.) [Figure A-3](#) shows the full version of this toolbar that is available in the layout window.

Figure A-3 Full Mouse Tools Toolbar

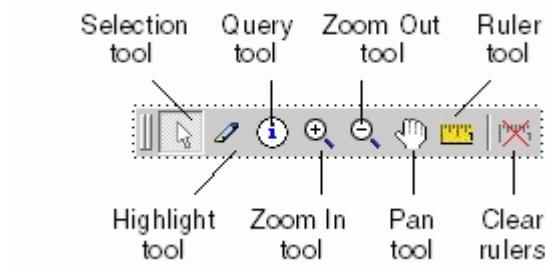


Figure A-4 shows the abbreviated version of the toolbar that is available in the main window, the timing analysis window, and the interactive CTS window.

Figure A-4 Abbreviated Mouse Tools Toolbar

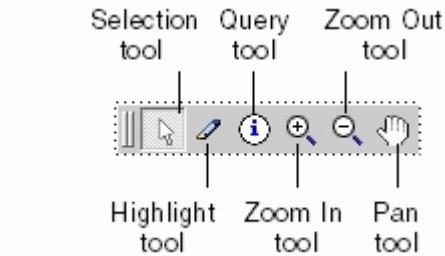


Table A-1 describes the tasks you can perform with the tools on the Mouse Tools toolbar.

Table A-1 Mouse Tools

To perform this task	Use this tool
To set the left mouse button to select objects by clicking or dragging with the Selection tool,	Click the button or choose View > Mouse Tools > Selection tool.
To set the left mouse button to highlight objects by clicking or dragging with the Highlight tool,	Click the button or choose View > Mouse Tools > Highlight Tool.
To set the left mouse button to display object information by clicking objects with the Query tool,	Click the button or choose View > Mouse Tools > Query Tool.
To set the left mouse button to magnify the design by clicking or dragging with the Zoom In tool,	Click the button or choose View > Mouse Tools > Zoom In Tool.
To set the left mouse button to shrink the design by clicking or dragging with the Zoom Out tool,	Click the button or choose View > Mouse Tools > Zoom Out Tool.
To set the left mouse button to scroll or navigate through the design by dragging the Pan tool,	Click the button or choose View > Mouse Tools > Pan Tool.
To set the left mouse button to draw rulers by clicking with the Ruler tool,	Click the button or choose View > Mouse Tools > Rule Tool.

Note:

IC Compiler also provides mouse tools that you can use to perform various layout editing tasks. For details about these mouse tools, see “[Editing Objects Interactively](#)” on [page A-107](#).

The Selection tool is the default mouse tool. When you open a layout window or a new layout view, schematic view, or histogram view, the Selection tool is the active mouse tool. When you enable a mouse tool in the active view, the tool remains active until you enable a different mouse tool. When you change the mouse tool, you change it only in the active view.

When you enable a mouse tool in a layout view, the options for using the tool appear on the Mouse Tool Options toolbar. See [Figure A-5 on page A-31](#) for an example of this toolbar. If you need assistance using the active mouse tool, you can click the  button to display a Help page in the man page viewer.

You can use the Zoom In tool, the Zoom Out tool, or the Pan tool as nested tools when you are performing an operation with another mouse tool. When you finish the nested tool operation, the left mouse button returns to the previous tool and you can resume the operation that you were performing with that tool.

For more information about using mouse tools, see the “Selecting a Mouse Tool” topic in IC Compiler Help.

Previewing Objects

You can preview objects in a layout, schematic, or clock graph view before you interact with them by using a mouse tool. For example, if you need to click an object in an area of the design where objects are densely packed or overlapped, you can preview the objects at a location before clicking the object you need. This preview mechanism is available for the Selection, Highlight, and Query tools in layout, schematic, and clock graph views and for the route editing tools in layout views.

When you move the pointer over an object, the tool displays the object in the preview color, which is white by default and has thinner lines than selection coloring. If InfoTips are enabled, the tool displays object information in an InfoTip. This information can include the object name, the object type, and some attribute values.

InfoTips are enabled by default in layout views and disabled by default in schematic views. You can disable InfoTips if, for example, you do not want to view them while traversing the design.

To enable or disable InfoTips in the active view,

- Choose View > InfoTips.

A check mark appears on the menu beside the InfoTips command when InfoTips are enabled.

In a layout view, if more than one object occupies the same location, the tool displays information about the object with the closest edge to the pointer. You can display the preview information sequentially for each object.

- To preview the next object in the sequence, press F1.
- To preview the previous object in the sequence, press Shift-F1.

Alternatively, you can move the pointer around the overlapping objects; when the InfoTip displays the information for the object you need, you can click the object.

For tools that operate on selected objects, you can preview any selectable object that is enabled for selection on the View Settings panel. For other tools, such as the Query tool, you can preview any object that the tool can operate on regardless of whether the object can be selected.

Note:

If you have trouble previewing small objects, try magnifying the view. In a large design, some objects or object outlines are too small to preview at low magnification. For details, see [“Magnifying and Traversing the Design” on page A-37](#).

The information that the InfoTip displays varies depending on the type of object. You can add or remove attributes in the InfoTip for an object type by modifying the `LayoutInfoTipText` attribute group in the Attribute Group Manager dialog box. For details, see [“Managing Attribute Groups” on page A-50](#).

For more information, see the “Previewing Objects” topic in IC Compiler Help.

Selecting Objects

You can use the Selection tool and the Select menu commands to select objects or timing paths in the design.

- Use the Selection tool () to select objects interactively in the active layout or schematic view.
- Use commands on the Select menu to select objects by name or type, select timing paths, or select fanin or fanout objects.

The GUI displays selected objects in the selection color, white by default, in all open views in which the objects appear, and adds the object names to the selection list. To view the selection list, click the  button on the status bar or choose Select > Selection List. If you need to deselect all selected design objects and timing paths in the current design, choose Select > Clear.

When you use the Selection tool, you can fine-tune selections by adding or removing objects in existing selections. Before you can select an object in a layout view, it must be enabled for display and selection.

To activate the Selection tool,

- Click the  button on the Mouse Tools toolbar or choose View > Mouse Tools > Selection Tool.

If you need to select an object at a location where objects are densely packed or overlapped, you can preview the objects at the location. In a layout view, you can preview multiple objects sequentially at the same location, forward or backward, by pressing F1 or Shift-F1. For more details, see [“Previewing Objects” on page A-28](#).

You can select or deselect objects in a schematic view or clock graph view, and by default in a layout view, by clicking or dragging the pointer.

- To select an object and deselect any other selected objects, click the object.
- To select multiple objects at the same time and deselect any other selected objects, drag the pointer to draw a rectangular box around the objects.

To add objects to the current selection, press the Ctrl key while you click or drag the pointer. To remove an object from the current selection, press the Shift key while you click or drag the pointer. In a layout view, you can set an option to control whether the tool replaces the current selection, adds objects to the current selection, or removes objects from the current selection.

Note:

If you have trouble selecting small objects, try magnifying the view. In a large design, some objects or object outlines are too small to click at low magnification. For details, see [“Magnifying and Traversing the Design” on page A-37](#).

To deselect all selected objects,

- Choose Select > Clear.

Alternatively, in a layout view, you can click the Clear button on the Mouse Tool Options toolbar.

In a layout view, you can fine-tune the Selection tool behavior by setting options on the Mouse Tool Options toolbar. [Figure A-5](#) shows the default options when the Selection tool is the active mouse tool. For Help adjust the Selection tool and its options in a layout view, click the  button on the Mouse Tool Options toolbar.

Figure A-5 Selection Tool Options on Mouse Tool Options Toolbar



- To select or deselect a single object, by clicking the object, or the objects in a rectangular area, by dragging the pointer over the objects, select the Smart option or press P. The Smart option is selected by default.
- To select or deselect objects in a rectangular area by either dragging the pointer or clicking two diagonally opposite corners of the rectangle, select the Rectangle option or press W. This allows you to click one corner of the rectangle and then use the nested zoom and pan tools before clicking the other corner.
- To select or deselect objects along a straight line by dragging across an edge of each object, select the Line option or press L.
- To select or deselect objects that are partially outside a rectangular area when the Smart or Rectangle input mode option is selected, select the Enable option.
- To control whether the tool replaces objects in, adds objects to, or removes objects from the current selection, select an option in the Selection list.

The choices are Replace, Add, and Remove. The default is Replace.

Viewing Object Information

You can use the Query Tool to view design and timing information for objects in a layout or schematic view. When you click an object in the active view, the information appears on the Query panel (the panel opens automatically if it is not already open). You can also set options on the Query panel to automatically or manually copy the information to the session transcript in the console log view. You can also control which attributes the Query panel displays.

To enable object queries in the active layout or schematic view,

- Click the button on the Mouse Tools toolbar (or choose View > Mouse Tools > Query Tool).

The Query panel appears, docked by default to the right side of the window. You should dock the panel or move it to a place on the screen where you can work with both it and the active view at the same time.

If you need to query an object at a location where objects are densely packed or overlapped, you can preview the objects at the location. In a layout view, you can preview multiple objects

sequentially at the same location, forward or backward, by pressing F1 or Shift-F1. For more details, see “[Previewing Objects](#)” on page A-28.

To display information for an object,

1. Make sure the  button is selected on the Mouse Tools toolbar.
2. Click the object.

The object is highlighted in the query color, which is white by default, and the information for the object appears on the Query panel. Query highlighting has thinner lines than selection highlighting.

Note that you can select and copy text on the Query panel, but you cannot edit the text.

Note:

If you have trouble querying small objects, try magnifying the view. In a large design, some objects or object outlines are too small to click at low magnification. For details, see “[Magnifying and Traversing the Design](#)” on page A-37.

To copy the object information into the session transcript in the console log view,

- Select the “Print to Log” option on the Query panel.

When this option is selected, information for objects you click in Query Tool is automatically copied to the transcript.

You can configure the information that appears on the Query panel by specifying which attributes the tool displays for different types of queries—single object query or multiple object query—and different types of objects. The default attribute groups for most object types are QueryText for single object queries and Basic for multiple object queries. You can add or remove attributes in the QueryText attribute group. For details, see “[Managing Attribute Groups](#)” on page A-50.

For object types that do not have a QueryText or Basic attribute group, the default attribute group is All. If you need to restore the dialog box options to their defaults, click Default. The All attribute group is available for every object and query type. The other groups that are available vary depending on the object and query type. You can select any attribute group that appears in the list for a particular object and query type.

For more details about using the Query tool and the Query panel, see the “Viewing Object Information” topic in IC Compiler Help.

Highlighting Objects

You can interactively highlight objects in the active layout or schematic view by using the Highlight tool. You can highlight individual objects or objects within a region. In a layout view, you can set options to highlight an object's net connections and to display information about an object on the Query panel. You can also remove highlighting from objects by using the Highlight tool.

When you highlight an object, the GUI displays the object in the current highlight color, which is yellow by default. In a layout view, if you enable highlighting for net connections, flylines appear in the highlight color. The GUI also highlights the object in all other open layout and schematic views.

To activate the Highlight tool,

- Click the  button on the Mouse Tools toolbar (or choose View > Mouse Tools > Highlight Tool).

When you activate the Highlight tool in a layout view, the Highlight tool options appear on the Mouse Tool Options toolbar.

- To display object information on the Query Object panel when you highlight an object, select the Query On Highlight option.
- To highlight the nets connected to objects that you highlight, select the Highlight Nets option, and select the options for the types of nets you want to highlight. The choices are Signal, Clock, Power, Ground, Tie High, and Tie Low. By default, Signal and clock are enabled, and the other options are disabled.

If you need to highlight an object at a location where objects are densely packed or overlapped, you can preview the objects at the location. In a layout view, you can preview multiple objects sequentially at the same location, forward or backward, by pressing F1 or Shift-F1. For more details, see “[Previewing Objects](#)” on page A-28.

You can use the Highlight toolbar and the Highlight menu to change the highlight color. By default, the GUI applies the same color to each object you highlight until you change the color. You can enable an automatic color cycling mechanism.

To enable or disable autocycling,

- Choose Highlight > Auto Cycle Colors.

A check mark next to the Auto Cycle Colors command on the Highlight menu indicates that autocycling is enabled.

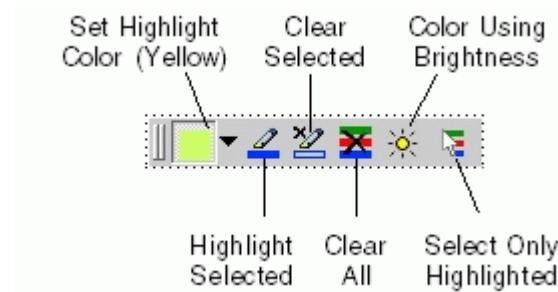
When you enable the automatic color cycling mechanism, the GUI automatically cycles through the highlight colors. Each time you apply highlighting to selected objects or paths, the GUI uses the next highlight color in the cycle.

To highlight objects by using the Highlight tool,

1. Make sure the  button is selected on the Mouse Tools toolbar.
 2. (Optional) Select a highlight color in the  list on the Highlight toolbar, or click the  button to highlight objects by brightness (non-highlighted objects are dimmed) instead of color.
 3. Highlight objects or remove highlighting in any of the following ways:
 - To highlight individual objects, click the objects.
- Note:**
- If you have trouble highlighting small objects, try magnifying the view. In a large design, some objects or object outlines are too small to click at low magnification. For details, see “[Magnifying and Traversing the Design](#)” on page A-37.
- To highlight objects in a rectangular area, drag the pointer diagonally to draw a box around the area. The GUI highlights objects that are completely inside the box.
 - To remove highlighting from individual objects, Ctrl-click the objects.
 - To remove highlighting from objects in a rectangular area, press Shift and drag the pointer diagonally to draw a box around the area. The GUI removes highlighting from objects that are completely inside the box.
4. Repeat step 2 or steps 2 and 3 as needed.

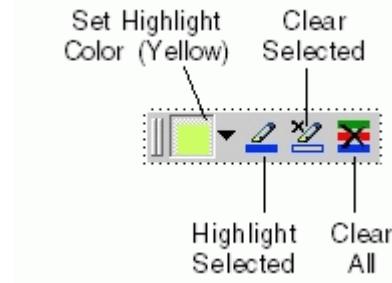
The Highlight toolbar lets you select the highlight color, add or remove highlighting on selected objects, remove highlighting on all objects, or highlight objects by brightness instead of color. This toolbar can be found in the main, layout, timing analysis, and interactive clock tree synthesis windows; however, the highlight by brightness capability is only available in layout windows. [Figure A-7](#) shows the full toolbar as it appears in the layout window.

Figure A-6 Full Highlight Toolbar



[Figure A-7](#) shows the abbreviated version of the toolbar that is available in the main window and the timing analysis window.

Figure A-7 Abbreviated Highlight Toolbar



You can also highlight selected timing paths in the active layout or schematic view. When you highlight a timing path, the pins and ports on the path appear in a different color from the unhighlighted objects. Flylines appear in the highlight color to show the connections between the objects on the path.

To highlight timing paths,

1. Select the paths you want to highlight.
2. Click the  button on the Highlight toolbar or choose Highlight > Selected.

You can remove the highlighting from selected objects or timing paths, remove the current highlight color from objects and paths highlighted with that color, or remove all highlighting in the active layout or schematic view.

To remove highlighting from selected objects and paths,

1. Select the objects and paths from which you want to remove the highlight color.
2. Click the  button on the Highlight toolbar or choose Highlight > Clear Selected.

To remove the current highlight color from objects and paths,

- Choose Highlight > Clear Current Color.

To remove all highlighting from objects and paths,

- Click the  button on the Highlight toolbar or choose Highlight > Clear All.

You can also enable a mechanism to select only those objects that are currently highlighted when you select objects interactively with the Selection tool or an editing mouse tool. For details, see [“Selecting Only the Highlighted Objects” on page A-36](#).

Selecting Only the Highlighted Objects

If you need to select a large number of objects incrementally, for example, or some but not all of the objects in an area of the design, you can enable a layout view option that permits the Selection tool and the editing mouse tools to select only highlighted objects. If you enable this option and then drag the Selection tool or an editing tool to select objects in a rectangular area, the tool selects only those objects that are currently shown in highlight coloring.

To enable the constraint to select only highlighted objects,

- Click the  button on the Highlight toolbar or choose Highlight > Select Only Highlighted.
A check mark appears next to the command on the Highlight menu when the capability is enabled.

When the View Settings panel is visible, you can select the “Only select highlighted objects” option and click Apply. A check mark appears on the option when the constraint is enabled

Selecting or Highlighting Objects by Name

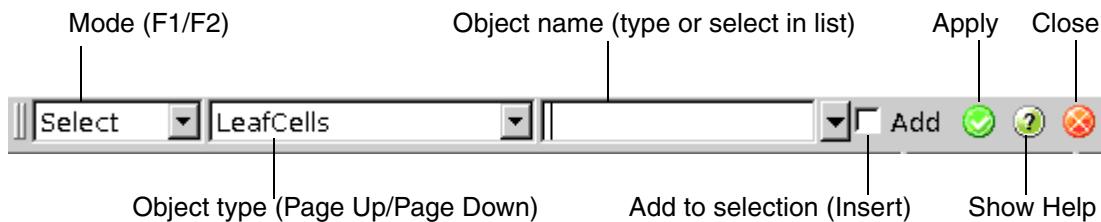
You can select or highlight objects in the active layout view by typing their names on the Select By Name tool. You can operate this tool by pressing keys on the keyboard or by setting options and typing the object names on the Select By Name toolbar.

Note:

The Select By Name toolbar is designed for fast object searches based on names or name patterns that you type. For more sophisticated object searches using reusable filters or regular expressions, use the Select By Name dialog box. For details, see [“Searching for Objects by Name” on page A-51](#).

To display the Select By Name toolbar,

- Choose Select > By Name Toolbar or press Ctrl+/.
The Select By Name toolbar appears below the layout view by default.

Figure A-8 Select By Name Toolbar

When you display the Select By Name toolbar, the keyboard focus automatically changes to the object name box on the toolbar. To select a cell by default, type its name and press Return. You can

- Cycle through the object type list by pressing the Page Up key or Page Down key.
- Type one or more object names, or select the names in the name completion list.
- Set the operating mode by pressing F1 to select objects or F2 to highlight objects.
- Press the Insert key to change between the add and replace selection operations.
- Press Enter to apply the selection or highlighting to the objects.

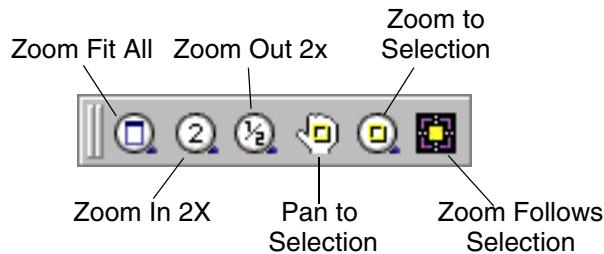
When you begin typing a name, the letters appear in the object name box on the toolbar. You can type multiple object names by separating them with blank spaces. You can allow the tool to complete a name you are typing by pressing Tab. The tool completes the name to its longest match. If the tool finds multiple objects with names that match, an object list appears showing the first 15 names.

For more information about the Select By Name toolbar, see the “Selecting Objects by Name” topic in IC Compiler Help.

Magnifying and Traversing the Design

You can use the zoom and pan tools and the zoom commands to find and display information in different parts of the design.

The View Zoom/Pan toolbar, shown in [Figure A-9](#), lets you instantly zoom or pan in increments defined by the tool. For example, you can zoom to a full view of the design, zoom in or out by a factor of 2, pan or zoom to a selection, or have zoom follow your selections. This toolbar can be found in the main, layout, timing analysis, and interactive clock tree synthesis windows.

Figure A-9 View Zoom/Pan Toolbar

You can magnify a view in any of the following ways:

- To double the magnification in the active view, click the button on the View Zoom/Pan toolbar or choose View > Zoom > Zoom In.
- To recenter the active view and double the magnification, click the button on the Mouse Tools toolbar or choose View > Mouse Tools > Zoom In Tool, and click where you want to center the view.
- To magnify a rectangular area to fill the active view, click the button on the Mouse Tools toolbar or choose View > Mouse Tools > Zoom In Tool, and drag the pointer diagonally across the area you need to magnify.

You can shrink a view in any of the following ways:

- To fit the entire design in the active view, click the button on the View Zoom/Pan toolbar or choose View > Zoom > Zoom Fit All.
- To shrink the active view to half the magnification, click the button on the View Zoom/Pan toolbar or choose View > Zoom > Zoom Out.
- To recenter the active view and shrink it to half the magnification, click the button on the Mouse Tools toolbar or choose View > Mouse Tools > Zoom Out Tool, and click where you want to center the view.
- To shrink the active view by filling a rectangular area, click the button on the Mouse Tools toolbar or choose View > Mouse Tools > Zoom Out Tool, and drag the pointer diagonally across the area you need to fill.
- To adjust the magnification in the active view to fit all selected objects that are visible in the view, click the button on the View Zoom/Pan toolbar or choose View > Zoom > Zoom to Selection.
- To adjust the magnification in the active view to fit all highlighted objects that are visible in the view, choose View > Zoom > Zoom Fit Highlight.

To magnify a location in the active layout view by specifying its coordinates,

1. Choose View > Zoom > Zoom To.

The Zoom To dialog box appears.

2. Type the coordinates for two corners of a rectangular area, top-left and bottom-right or bottom-left and top-right, in the Coordinates box.

3. Click OK or Apply.

When only part of the design is visible in the active layout or schematic view or part of the histogram is visible in the active histogram view, you can use the Pan tool or the scroll bars to traverse the view to see different areas of the design or histogram. You can also use the pan commands to recenter the design by displaying selected objects or highlighted objects in the active layout or schematic view.

You can traverse a view in any of the following ways:

- To traverse the active view in any direction, click the  button on the Mouse Tools toolbar or choose View > Mouse Tools > Pan Tool, and drag the pointer in the direction that you want to move the design within the view.
- To recenter the active view and display selected objects, click the  button on the View Zoom/Pan toolbar or choose View > Zoom > Pan to Selection.
- To recenter the active view and display highlighted objects, choose View > Zoom > Pan to Highlight.
- To apply the magnification level in the active layout view to other open layout views in the same layout window, choose View > Mouse Tools > Zoom Layout View to Current View.
- To shrink the active view by filling a rectangular area, click the  button on the Mouse Tools toolbar or choose View > Mouse Tools > Zoom Out Tool, and drag the pointer diagonally across the area you need to fill.
- To traverse a view up or down by using vertical scroll bar, click the top or bottom scroll arrows or drag the scroll box.
- To traverse a view left or right by using the horizontal scroll bar, click the left or right scroll arrows or drag the scroll box.

The scroll boxes represent the visible part of the design. When you move a scroll box, the view moves in that direction.

You can also enable a mechanism that automatically zooms to an object when it is selected, and you can reuse zoom and pan settings. For more information, see the following sections:

- [Following Selected Objects](#)
- [Reusing Zoom and Pan Settings](#)
- [Redrawing the Active View](#)

Following Selected Objects

You can control whether a layout or schematic view automatically zooms to an object when you select it, thus increasing or decreasing the magnification of the design to fit the object within the view.

You can enable or disable this “follow-selection” mechanism separately for each open view. When you enable follow selection in a view, it remains enabled even when the view is not the active view. Follow selection is disabled by default.

To enable or disable the follow-selection mechanism for the active layout or schematic view,

- Click the  button on the View Zoom/Pan toolbar or choose View > Zoom > Follow Selection.

A check mark next to the Follow Selection command on the Zoom menu indicates that the follow-selection mechanism is enabled.

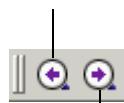
Reusing Zoom and Pan Settings

The IC Compiler GUI maintains a list of zoom and pan settings for each layout or schematic view. Each zoom and pan setting consists of both a zoom (magnification) level and a pan position (visible area). When you change the magnification or traverse the view (by panning or scrolling), the previous zoom and pan setting is added to the list.

The Zoom and Pan History toolbar, shown in [Figure A-10](#), lets you use previously used or specially defined zooms. It also lets you define special zoom settings for future use. This toolbar can be found in the main and layout windows.

Figure A-10 Zoom and Pan History Toolbar

Go Back in
Zoom and Pan History



Go Forward in
Zoom and Pan History

You can

- Reverse the zoom level and pan position to the last zoom and pan setting by stepping backward in the list toward the initial display you encountered when you opened the view
- Reapply the next (previously reversed) zoom level and pan position by stepping forward in the list after stepping backward in the list
- Save the current zoom level and pan position in a list of named zoom and pan settings
- Return the view to a certain zoom level and pan position by selecting a named zoom and pan setting that you saved earlier in the session

For more information, see the “Reusing Zoom and Pan Settings” topic in IC Compiler Help.

Redrawing the Active View

Certain operations (such as zooming or panning) in a layout view cause the GUI to automatically redraw the view. In addition, you can manually redraw the active layout or schematic view at any time.

To redraw the active view,

- Choose View > Refresh.

If necessary, you can recompute the boundary of the entire design, including the die area, to fit within the active layout view.

To recompute the boundary of the entire design,

1. Choose View > Zoom > Recompute Cell Boundary.

The GUI recalculates the cell boundary coordinates.

2. Update the layout view by choosing View > Zoom > Zoom Fit All.

Examining Design and Object Information

In addition to graphic views, the IC Compiler GUI provides tools and commands that you can use to select and examine design objects and other design data. You can view and edit object properties, explore the design hierarchy, view reports, and select or search for objects that you want to select or highlight.

You can save an image of a graphic view that you can view or print later outside of IC Compiler. The GUI provides a Tcl command that you can use to save an image of any open top-level GUI window or view window.

For information about performing these tasks, see the following sections:

- [Viewing and Editing Object Properties](#)
- [Exploring the Design Hierarchy](#)
- [Exploring the Virtual Hierarchy of a Flat Design](#)
- [Managing Attribute Groups](#)
- [Searching for Objects by Name](#)
- [Selecting and Viewing Timing Paths](#)
- [Viewing Reports](#)
- [Saving an Image of a Window or View](#)

Viewing and Editing Object Properties

You can use the Properties dialog box to view and edit attributes and other properties for selected designs, design objects, or timing paths. For example, to view properties for a cell that you see in a layout or schematic view, you can select the cell and then choose Edit > Properties. You can also set, change, or remove the values for some of the properties.

The Properties dialog box lists the properties in a table with two columns: property names and property values. The properties you can view include object names, attribute values, and certain timing and placement values. The list of properties differs depending on the types of objects that you select in the design and the attribute group that you select in the Properties dialog box.

Note:

Timing attribute values do not appear until you perform an operation that updates timing information, such as generating a timing report or opening a histogram.

To open the Properties dialog box,

1. Select one or more objects.
2. Choose Edit > Properties.

The Properties dialog box appears, with the number of selected objects displayed above the property list table. The property values for the first selected object appear in the table.

Note:

The Properties dialog box displays properties for the objects that are currently selected. If you change the current selection when the Properties dialog box is open, the dialog box changes to display the properties for the newly selected objects.

You can control which attributes appear in the Properties dialog box. The attributes are organized into attribute groups. For most object types, you can display all attributes, all application (predefined) attributes, all user-defined attributes, or just the basic (most frequently used) attributes. For cells, you can also display placement or timing attributes. In addition, for types of objects that can be previewed or queried in the layout view, you can display the attributes that appear in the preview InfoTip or on the Query panel.

To control which attributes appear in the Properties dialog box,

- Select an option in the “Attribute group” list.

The default attribute group depends on which type of object you select. If you select multiple objects of different types, only the attributes in the basic attribute group appear.

You can modify the contents of some attribute groups and create custom, user-defined attribute groups by using the Attribute Group Manager dialog box.

To open Attribute Group Manager dialog box,

- Click the  button in the Properties dialog box.

For more details about attribute groups, see [“Managing Attribute Groups” on page A-50](#).

When you select multiple objects, the property lists are displayed separately for each object.

You can use the previous and next arrow buttons ( and ) to navigate from one list of object properties to another.

You can select an option to list the property values for all the selected objects together in a single table.

- If you try to display properties for all objects of the same type, the dialog box shows the values that are identical for all the objects and displays “<Multiple values>” for other values.
- If you try to display properties for all objects of more than one type, the dialog box does not display any properties.

For more information about using the Properties dialog box, see the “Viewing and Editing Object Properties” topic in IC Compiler Help.

Exploring the Design Hierarchy

You can use the hierarchy browser to explore the design hierarchy and examine the blocks and hierarchical cells in the design. If you are not familiar with a design, you can explore the hierarchy to understand its structure and gather information about the design objects in a block or hierarchical cell. Use the hierarchy browser to

- Explore the complete hierarchical structure of the design and observe how many hierarchical blocks are present and the size of each block
- View design information for the objects (cells, ports and pins, or nets) within a block or hierarchical cell

You can select the hierarchical cells, leaf cells, or other objects that you want to examine in a layout or schematic view or with other analysis tools.

- Examine hierarchical cell placement in the layout view

You can map the leaf cells descended from each hierarchical cell by applying colors to the hierarchical cells in the hierarchy browser.

To open the hierarchy browser in the main window,

- Choose Hierarchy > New Hierarchy Browser View.

To open the hierarchy browser in the layout window,

- Choose Partition > New Hierarchy Browser View.

The Partition menu is available only when the current task in the layout window is set for design planning (choose File > Task > Design Planning) or all tasks (choose File > Task > All Tasks).

The hierarchy browser view window consists of two panes, with an instance tree view that is always visible and an object list view that is hidden by default. The instance name of the top-level design appears at the top of the instance tree. The icons beside the cell names represent their cell types (standard cells, hard or soft macros, and so forth).

To display or hide the object list view,

- Right-click in the instance tree view and choose Show Child List.

A check mark next to the Show Child List command indicates that cell list is visible.

The object list view contains an object table that displays details about the cells, pins and ports, or nets in the selected block or hierarchical cell.

Examining Hierarchical Cells

To explore the design hierarchy and examine the design objects within each hierarchical cell, you can

- Expand hierarchical cells, showing the names of the instances at the next level in the hierarchy
To expand a hierarchical cell, click the expansion button (plus sign) next to the instance name.
- Select hierarchical cells in the instance tree view to display information about leaf cells or other objects in the object list view
You can filter the cells by type and display or hide columns in the instance tree or the object table.
- Select an object type in the list above the object table to display object information for cells, ports and pins, or nets
The object table displays cell information by default.

You can select an object in the hierarchy browser by clicking its row in the instance tree or the object table. You can select multiple objects by dragging the pointer over their rows, and you can Shift-click or Ctrl-click rows to select combinations of objects.

For more information about using the hierarchy browser, see the “Exploring the Design Hierarchy” topic in IC Compiler Help.

Examining Hierarchical Cell Placement

You can examine hierarchical cell placement in the layout view by applying colors to cells in the hierarchy browser. You can color complete cell hierarchies, selected hierarchical cells, or selected leaf cells. The layout view highlights leaf cells with the applied colors. The hierarchy browser displays the cell colors on the icons beside the cell names.

To automatically assign colors to top-level hierarchical cells,

- Right-click and choose Set Top Hierarchy Colors.

To set the color for selected instances or leaf cells,

1. Select the cells.
2. Right-click and choose Set Selected Hierarchy Color to open the Auto Color menu.
3. Choose a color.

For more information about coloring hierarchical cells, see the “Examining Hierarchical Cell Placement” topic in IC Compiler Help.

Exploring the Virtual Hierarchy of a Flat Design

You can explore the virtual hierarchy of a flat design and examine the virtual hierarchical cells in the design. If you are not familiar with a design, you can explore the virtual design hierarchy to understand its structure and gather information about the leaf cells in each virtual hierarchical cell.

The Flat Design Hierarchy Browser dialog box creates and displays a virtual design hierarchy based on a delimiter that you specify. You can

- Explore the complete virtual hierarchical structure of the flat design and observe how many virtual hierarchical cells are present and the size of each cell
- Examine design information for the leaf cells within a virtual hierarchical cell

You can select the leaf cells that you want to examine in a layout or schematic view or with other analysis tools.

- Examine the virtual hierarchical cell placement in the layout view
You can map the leaf cells descended from each virtual hierarchical cell by applying colors to the virtual hierarchical cells in the Flat Design Hierarchy Browser dialog box.

Use the Flat Design Hierarchy Browser dialog box if you need to debug the quality of placement for macro and standard cells, find critical placement-related problems, or examine the timing optimization of module boundary logic.

Note:

The Flat Design Hierarchy Browser dialog box is available only when the current design is a flat design. It is not available when the current design is a partially or fully hierarchical design.

To open the Flat Design Hierarchy Browser dialog box,

1. Choose one of the following commands:

- In the main window, choose Hierarchy > Flat Design Hierarchy Browser.
- In the layout window, choose Floorplan > Flat Design Hierarchy Browser.

The Flat Design Hierarchy Browser command appears on the Floorplan menu only when the current task in the layout window is set for block implementation (choose File > Task > Block Implementation) or all tasks (choose File > Task > All Tasks).

The Flat Design Hierarchy Browser - Separator dialog box appears.

2. (Optional) Type the character that you want the tool to use as the delimiter to build the virtual hierarchy tree from the flat design.

The default delimiter is a slash (/).

3. Click OK.

The Flat Design Hierarchy Browser dialog box contains two views: a virtual hierarchy tree view that is always visible and a cell list view that is hidden by default. The instance name of the top-level design appears at the top of the virtual hierarchy tree. The icons beside the cell names represent their cell types (standard cells, hard or soft macros, and so forth).

The cell list view contains an object table that displays details about the leaf cells in the selected virtual hierarchical cell.

To display or hide the cell list view,

- Right-click in the virtual hierarchy tree view and choose Show Child List.

A check mark next to the Show Child List command indicates that cell list is visible.

If you edit the design and change the netlist, you can regenerate the virtual design hierarchy by clicking the Reload button below the virtual hierarchy tree.

To close the Flat Design Hierarchy Browser dialog box,

- Click Close.

Examining Virtual Hierarchical Cells

To explore the virtual hierarchical structure of a flat design and examine the leaf cells within each virtual hierarchical cell, you can

- Expand virtual hierarchical cells, showing the names of the virtual hierarchical blocks or cells at the next level in the hierarchy

To expand a virtual hierarchical cell and view the next level in the virtual hierarchy, click the expansion button (plus sign) next to the cell name.

- Copy the hierarchy paths for virtual hierarchical cells

To copy a hierarchy path, select the hierarchical cell, and then right-click and choose Copy Hierarchy Path. You can paste the path on the command line, in a dialog box, or in an external tool such as a text editor.

- Select hierarchical cells in the virtual hierarchy tree to display leaf cell information in the cell list view

You can filter the leaf cells, display or hide cell attributes, generate histograms, and export the leaf cell information to a text file.

You can select a virtual hierarchical cell or a leaf cell by clicking its row in the virtual hierarchy tree or the object table. You can select multiple cells by dragging the pointer over their rows, and you can Shift-click or Ctrl-click rows to select combinations of cells.

Note:

Selecting a virtual hierarchical cell in the Flat Design Hierarchy Browser dialog box clears the global selection list.

To facilitate your examination of the leaf cells within a virtual hierarchical cell, you can control the information in the object table by displaying or hiding columns, generate histograms that show the distribution of values in specific columns, and save the leaf cell information by exporting it to a text file.

To control the object information in the cell list view,

1. Select an attribute group in the list below the object table.

The object table displays a column for each attribute in the attribute group. The default attribute group is Basic, which contains only the most frequently used cell attributes.

2. (Optional) Click the  button below the object table to open the Attribute Group Manager dialog box.

You can modify an attribute group or create a user-defined attribute group.

For more information about attribute groups, see “[Managing Attribute Groups](#)” on [page A-50](#). You can also filter the object table to view information for just the leaf cells in which you are interested. For details, see the “[Filtering Object Lists](#)” topic in IC Compiler Help.

To generate a histogram that shows the distribution of values in a column,

1. Click the Histogram button below the object table, or click the  button and choose Histogram.
2. Set options as needed in the Table Histogram dialog box and click OK.

For more details, see “[Viewing Histograms](#)” on [page A-87](#).

To save the leaf cell information in a text file,

1. Click the Export button below the object table, or click the  button and choose Export.
2. Select or type the file name in the Export to File dialog box and click Save.

For more information about using the Flat Design Hierarchy Browser dialog box, see the “[Exploring the Virtual Hierarchy of a Flat Design](#)” topic in IC Compiler Help.

Examining Virtual Hierarchical Cell Placement

You can examine the quality of cell placement in the layout view by applying colors to virtual hierarchical cells. The layout view displays the leaf cells with the applied colors and the Flat Design Hierarchy Browser dialog box displays the cell colors on the icons beside the cell names.

To apply colors automatically to the top-level cells in the virtual hierarchy tree,

- Right-click and choose Set Colors On Top Hierarchies.

The tool applies a different color to the leaf cells descending from each top-level hierarchical cell.

Alternatively, you can use the `gui_set_flat_hierarchy_color` command with the `-cycle_color` option.

To apply a certain color to the leaf cells descending from one or more virtual hierarchical cells,

1. Select the virtual hierarchical cell names in the hierarchy tree.
2. Right-click and choose Set Selected Hierarchy Color to open the color menu.
3. Choose a color.

If you choose Automatic, the tool applies the next color in the color cycle. If you choose More, the Library Colors dialog box appears. Select a color and click OK.

Alternatively, you can apply a color to a virtual hierarchical cell by using the `gui_set_flat_hierarchy_color` command with the `-color` option.

To remove colors from one or more virtual hierarchical cells,

1. Select the virtual hierarchical cell names in the hierarchy tree.
2. Right-click and choose Clear Selected Hierarchy Color.

To remove the colors from all the cells in the virtual hierarchy tree,

- Right-click and choose Clear All Hierarchy Colors.

Alternatively, you can remove the color from a virtual hierarchical cell by using the `gui_unset_flat_hierarchy_color` command.

Each time you apply or remove colors in the Flat Design Hierarchy Browser dialog box, the tool logs the equivalent `icc_shell` command and displays it in the console log view. When you save the design, the tool saves any colors that are currently applied to virtual hierarchical cells in the flat design hierarchy browser.

Managing Attribute Groups

Attribute groups are collections of attributes that the GUI uses to determine which attribute values to display in lists of design objects. The content of these groups varies depending on the object type.

You can control which attribute group the GUI uses to display attribute values when you

- View or edit object properties in the Properties dialog box
- View object information on the Query panel
- View selected objects in the Selection List dialog box
- View object information in object list views

You can view both application attributes that are defined in IC Compiler and user-defined attributes that you define by using the `define_user_attribute` command.

The GUI provides the following predefined attribute groups:

- The Application attribute group contains all the attributes for an object that are defined in IC Compiler.
- The User attribute group contains any user-defined attributes that you have defined for an object by using the `define_user_attribute` command.
- The All attribute group contains all the application and user-defined attributes for an object.
- The Basic attribute group contains only the most frequently used attributes for an object.
- The LayoutInfoTipText attribute group contains the attributes that appear in the InfoTip when you preview an object in the layout view.
- The QueryText attribute group contains the attributes that appear on the Query panel when you query an object with the Query tool.

In addition, the GUI provides placement, timing, path, floorplan, relative placement, and wire attribute groups for the appropriate types of objects.

The tool updates the All and User groups automatically when you create or remove a user-defined attribute.

You can use the Attribute Group Manager dialog box to view, modify, create, and remove attribute groups. You can create, rename, and remove user-defined attribute groups but not attribute groups that are defined in the GUI. You can modify any attribute group except All, Application, and User. For example, you can customize preview InfoTips in the layout view by modifying the LayoutInfoTipText attribute group. You can also copy an attribute group to

create a new user-defined attribute group with the same content, and then modify the group to meet your needs.

You can save attribute groups in your preferences file, load them from the preferences file, or reset the attribute groups to their system defaults.

To open the Attribute Group Manager dialog box,

- Click the  button in the Properties dialog box, the Selection List dialog box, or an object list view.
- On the Query panel, you can choose Options > Customize to open the Customize Query Toolbar dialog box, and then click the Attribute Groups button.

For more information about working with attribute groups, see the “Managing Attribute Groups” topic in IC Compiler Help.

Searching for Objects by Name

You can use the Select By Name dialog box to search for objects of any type by specifying a name or name pattern or by defining a regular expression. Then you can select or highlight some or all of the objects found in the search.

To search for objects, you select an object type and a design scope. You can search

- All objects in the design
- Selected objects
- All or selected objects found in the previous search

You can define and save filters for different object types based on object name patterns and attribute or function values. You can also copy or modify existing filters.

Note:

The Select By Name dialog box is designed for sophisticated object searches using filters or regular expressions. For faster object selection using object searches based on names or name patterns that you type, use the Select By Name toolbar. For details, see [“Selecting or Highlighting Objects by Name” on page A-36](#).

The search results include the total number of objects found and a list of object names with the values of any attributes or functions used to generate the search. When you are satisfied with the search results, you can select the names of objects that you want to select, deselect, or highlight in the active layout view.

To search for and select objects by name or regular expression,

1. Choose Select > By Name to open the Select By Name dialog box.
2. Select an object type and a design scope in the left and right “Search for” lists.
The default object type is Leafiest and the default design scope is “In entire design.”
3. Specify the name or regular expression by doing one of the following:
 - To search for objects by name, select the “Wildcard matching” option (the default), and type an object name or name pattern (with wildcards) in the Name box.
 - To search for objects by using a regular expression, select the “Full regular expressions” option, and either define a search filter or select the name of a previously defined filter in the Filter list.

You can define a search filter with one or more terms. To start a new filter, click the Add Term button and select an object name, an operator, and a value in the table below the Name box. To add a term, select AND or OR in the Next Term list.

If you want to save the filter for future use, click the Save button to open the Save Filter As dialog box, type a filter name, and click OK. The GUI adds the filter name to the Filter list and saves the filter definition in your preferences file. You can delete a saved filter by selecting the filter name in the Filter list and clicking the Delete button.

4. Set other options as needed.
5. Click Search.

The names of the objects found in the search appear in the “Search results” list. Initially, all the names in the list are selected. You can modify the list by removing object names or by performing another search and adding the object names to the list.

6. Select the names of objects in the “Search results” list that you want to use to select, deselect, or highlight in the most recently active layout view.
7. Select, deselect, or highlight the objects by doing one or more of the following:
 - To select objects and replace the current selection, select the object names, select the Replace option and click the Change Selection button.
 - To select objects and add them to the current selection, select the “Add to” option and click the Change Selection button.

- To deselect objects (remove them from the current selection, select the “Remove from” option and click the Change Selection button).
- To highlight objects in the most recently active layout view, click the Add to Highlight button.

You can also display the object names in a new list view by clicking the Show in Table View button.

8. To close the Select By Name dialog box, click Close.

Selected objects are displayed in the selection color (white) in graphic views and indicated by reverse video in hierarchy views, their names appear in the selection list, and the number of selected objects appears in the status bar. Highlighted objects appear in the current highlight color.

Selecting and Viewing Timing Paths

You can select timing paths that you want to view or highlight in a layout or schematic view. Use the Select Paths dialog box to select the paths with the worst slack in the design or in a path group. You can also select individual timing paths to or from specific inputs, outputs, or registers.

When you select paths with the worst slack in the design, you can

- Include specific paths
- Set the maximum number of paths in the design, in a specific path group, or to a single endpoint
- Set other timing path options

You can also specify the selection bus and the selection operation.

To select specific paths or the paths with the worst slack in the design,

1. Choose Select > Paths From/Through/To.

The Select Paths dialog box appears.

2. (Optional) Specify individual timing paths by entering the startpoint, throughpoint, or endpoint names necessary to identify the paths in the From, Through, or To boxes.

Only paths that begin at valid startpoints or end at valid endpoints are included in the selection. If you specify more than one startpoint or endpoint, all paths that start at any of the startpoints or end at any of the endpoints (and meet the other criteria) are included.

3. Set options to control the maximum number and type of paths in the selection.
 - To change the maximum number of paths to an endpoint, enter a value in the “Nworst paths” box.
 - To change the maximum number of paths, enter a value in the “Max paths” box.
 - To specify a path group, select its name in the “Group name” list.
4. Set other options as needed.
5. Click OK or Apply.

When you select a timing path, the pins and ports on the path appear in the selection color (white). Flylines appear in the selection color to show the connections between the objects on the path.

Viewing Reports

The GUI displays reports in report views. By default, when you generate a report by choosing a report command on a menu, the GUI displays the report in a new report view.

Note:

When you enter a report command on the command line, the tool displays the report in the console log view and in `icc_shell` but does not open a report view.

You can use report views to

- Save a report in a text file
- Open a report previously saved in a text file

You can also search for words or phrases in a report. You can remove report text from a report view and use the blank view to open a report from a text file.

To save a report displayed in a report view,

1. Click the  button.

The “Save Report to File” dialog box appears.

2. Navigate to the directory where you want to save the report.

3. Select a file name, or type a file name in the “File name” box.

If you select or enter the name of an existing file, the GUI overwrites the file.

4. Click Save.

To find a word or phrase in the report text,

1. Type the word or phrase in the find box near the top-right corner of the view.

2. Click the  button.

The report view highlights the first occurrence of the word or phrase.

3. (Optional) Repeat step 2 to find the next occurrence of the word or phrase in the report text.

To remove the report text from a report view,

- Click the  button.

To display a report saved in a file,

1. Click the  button.

The Open Report File dialog box appears.

2. Navigate to the directory where the report file is located.
3. Select the file name.
4. Click Open.

Saving an Image of a Window or View

You can capture a picture of the design in a layout view and save it in an image file. You can save the image in any of the following formats: BMP, JPEG, PNG, or XPM. You can save an image of entire layout window (the default) or just the layout view.

To save an image of the design in the active layout view,

1. Choose View > Save Screenshot As.

The Save Screenshot As dialog box appears.

2. Navigate to the directory where you want to save the image file and select or type a file name in the “File name” box.

You can save the image in BMP, JPEG, PNG, or XPM format by using the appropriate file name extension. The default format is PNG.

3. To save an image of just the layout view, select the “Grab screenshot of layout view window only” option.

By default, the image includes the entire layout window.

4. Click Save.

Alternatively, you can use the `gui_write_layout_image` command.

You can capture a picture of any open GUI window or view window and save it in an image file. The image format can be PNG (the default), BMP, JPEG, or XPM. The image shows the window exactly as it appears on the screen but without the window border or title bar. For example, if you save an image of the active schematic view, the image shows the visible portion of the schematic at the current zoom level and pan position.

To save an image of a GUI window or view window, use the `gui_write_window_image` command to specify the file name, image format, and window name. Window instance names appear in the window title bars and on the Window menu.

Use the `-file` option to specify the file name. This option is required. For example, to save a PNG image of the active schematic view in a file named `my_schematic.png`, you can enter

```
icc_shell> gui_write_window_image -file my_schematic
```

You can use a file name extension or the `-format` option to specify the image format. The default image format is PNG. For example, to save an XPM image of the active layout view in a file named `my_layout.xpm`, enter either of the following commands:

```
icc_shell> gui_write_window_image -file my_layout.xpm
icc_shell> gui_write_window_image -file my_layout -format xpm
```

Use the `-window` option to specify the window. For example, to save a PNG image of the layout window named `Layout.1` in a file named `mux_1.png`, enter either of the following commands:

```
icc_shell> gui_write_window_image -file mux_1 -window Layout.1
```

For more details about saving window and view images, see the “Saving an Image of a Window” topic in IC Compiler Help.

Viewing the Physical Design

The layout view displays a very accurate graphic representation of the physical design. You use the layout view to visually examine the floorplan, placement, clock tree, and routing objects and gather information that can help you work on the design. You can open multiple layout views to simultaneously work with different areas of the design.

To open a layout view,

- In the layout window, choose View > New Layout View.

You can use the Overview panel to magnify or traverse the design in the active layout view. If multiple layout views are open, you can change the active view to a different layout view.

Layout data can be densely packed with overlapping objects on different layers. You can control the visibility (display or hide) and selection (enable or disable) of individual object types or layers by setting options on the View Settings panel. You can also customize object or layer properties (color, fill pattern, and so forth) and set other layout and object display options. If you open multiple layout views, you can set different options for each view.

Typically you set layout view properties for each design. For example, the colors you use for individual layers might change depending on the number of layers in the design. If you change settings in the active view and want to use the same settings in another view or during a future session, you can save them in your preferences file. You can also restore previously saved view settings by loading them from your preferences file.

You can visualize the physical layout as explained in the following sections:

- [Navigating Through Layout Views](#)
- [Displaying Grid Lines](#)
- [Drawing Rulers](#)
- [Adjusting the Color Brightness](#)
- [Displaying Cell Orientations](#)
- [Setting and Saving View Properties](#)
- [Viewing Design Overlays](#)
- [Using Stroke Activated Commands](#)

Navigating Through Layout Views

In the layout window, the Overview panel shows you what portion of the design is visible in each open layout view.

- The portion of the design displayed in the active layout view is shown as a solid yellow rectangle.
- The portions of the design displayed in other layout views are shown as solid gray rectangles.

You can use the Overview panel to quickly magnify or traverse the design in the active layout view. When multiple layout views are open, you can quickly change the active view to a different layout view.

By default, the Overview panel appears docked below the toolbars at the top left corner of the layout window. If you hide the Overview panel, you can redisplay it by choosing *View > Toolbars > Overview*.

For more information, see the “navigating Through Layout Views” topic in IC Compiler Help.

Displaying Grid Lines

You can display or hide the lithography grid and the user grid. By default, both grids are hidden.

To display or hide the lithography grid,

- Choose View > Show Litho Grid.

To display or hide the user grid,

- Choose View > Show User Grid.

To switch between the default grid spacing and ten times the default grid spacing,

- Choose View > Cycle Grid Spacing.

Drawing Rulers

You measure distances in a layout view by drawing rulers. A ruler can be composed of one or more horizontal, vertical, or diagonal segments. By default, the distances from the beginning of the ruler are labeled at the ends of each segment and at every tenth tick mark within a segment.

While you draw a ruler, the GUI displays a ghost image of the ruler in the drag object color (red by default), and displays the coordinates for the current point position on the status bar. The rulers also appear in any other layout views that you have open in the layout window.

Note:

You cannot reverse (by choosing Edit > Undo) ruler drawing operations.

To draw rulers in a layout view,

1. Click the  button on the Mouse Tools toolbar or choose View > Mouse Tools > Ruler Tool.

When you enable the Ruler tool in a layout view, the Ruler Tool options appear on the Mouse Tool Options toolbar and the pointer changes shape. By default, cross hair rulers appear attached to the pointer in the layout view.

2. (Optional) Set options as needed on the Mouse Tool Options toolbar.

You can

- Select a direction option
- Display or hide the cross hair rulers attached to the pointer when you start a new ruler. The labels that show the distance from the ruler origin to the endpoint of each ruler segment, or the labels that show the segment length at the midpoint of each ruler segment
- Control whether the distances displayed by the ruler labels are absolute measurements from the design origin, relative measurements from the ruler origin, or relative measurements from each segment origin
- Change the grid snapping option for the ruler segments
- Record the ruler segment lengths or distances in the session transcript (console log view)

3. Click in the layout view where you want to begin the ruler.

A ghost line appears indicating the horizontal or vertical distance between the initial point and the current pointer location.

4. Click where you want to end the ruler segment.

The ruler segment appears.

5. (Optional) To draw another ruler segment, repeat steps 2 through 4.

6. To end the ruler, press the Esc key or right-click and choose End Ruler.

You can also end a ruler by selecting a different mouse tool (click a button on the Mouse Tools toolbar).

7. (Optional) To draw another ruler, repeat steps 2 through 6.

You can remove individual rulers or all rulers in the layout view, but you cannot remove or modify individual ruler segments.

To remove a ruler from the layout view,

1. Move the pointer over the ruler you need to remove.
2. Right-click and choose Remove Nearest Ruler.

To remove all rulers from the layout view,

- Click the  button on the Mouse Tools toolbar (or choose View > Mouse Tools > Clear Rulers).

For more information about using interactive mouse tools, see “[Selecting a Mouse Tool](#)” on page [A-26](#).

Adjusting the Color Brightness

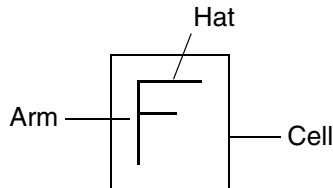
When you activate a visual mode or a map mode, the GUI dims the visible objects in the layout view. You can use the Brightness option on the View Settings panel to control the visual contrast between the visual mode or map mode colors and the other visible objects in the layout view. The GUI automatically dims the visible objects by resetting the color brightness to 33 percent.

To adjust the contrast between the color overlay and visible objects in the layout view,

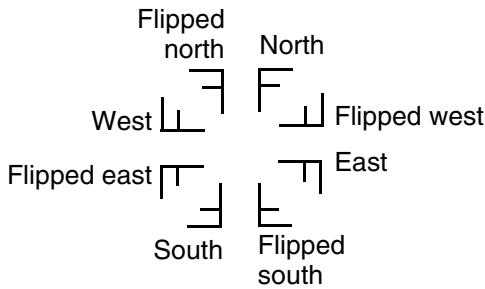
- On the View Settings panel, select a value in the Brightness list.
The choices are 100, 75, 66, 50, 33, 29, and Off.

Displaying Cell Orientations

IC Compiler represents cell orientation in the layout view as an F. The position of the “hat” of the F indicates the first direction; the “arm” indicates the second direction.



Cells can be oriented in any of the following ways:



To display cell orientations in the active layout view,

1. On the View Settings panel, click the Settings tab.
2. Click the Cells tab.
3. Select or deselect the Show Cell Orientation option.
4. Click Apply.

You can change the orientations of individual cells. For details, see the “Changing Cell Orientations” topic in IC Compiler Help.

Setting and Saving View Properties

The View Settings panel provides options you that can use to set display properties in the active layout view. You can also save the current settings in your preferences file, or load settings from the preferences file. If you open multiple layout views, you can set different options for each view.

To display or hide the View Settings panel,

- Choose View > Toolbars > View Settings.

A check mark beside the command on the Toolbars menu indicates that the View Settings panel is visible.

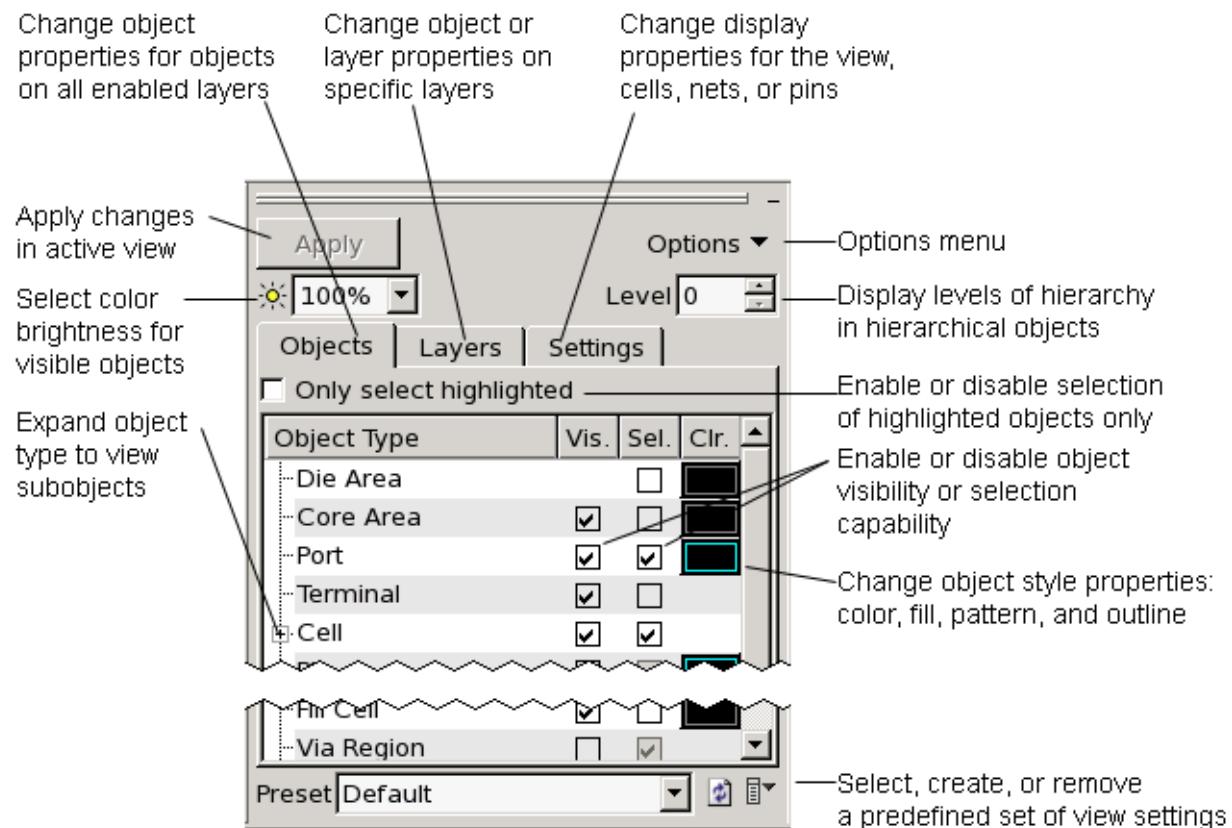
You can set options on the View Settings panel to

- Control object or layer visibility and selection
- Change object or layer style properties such as colors or fill patterns
- Set display options for the view or for cells, ports, pins, or nets in the view

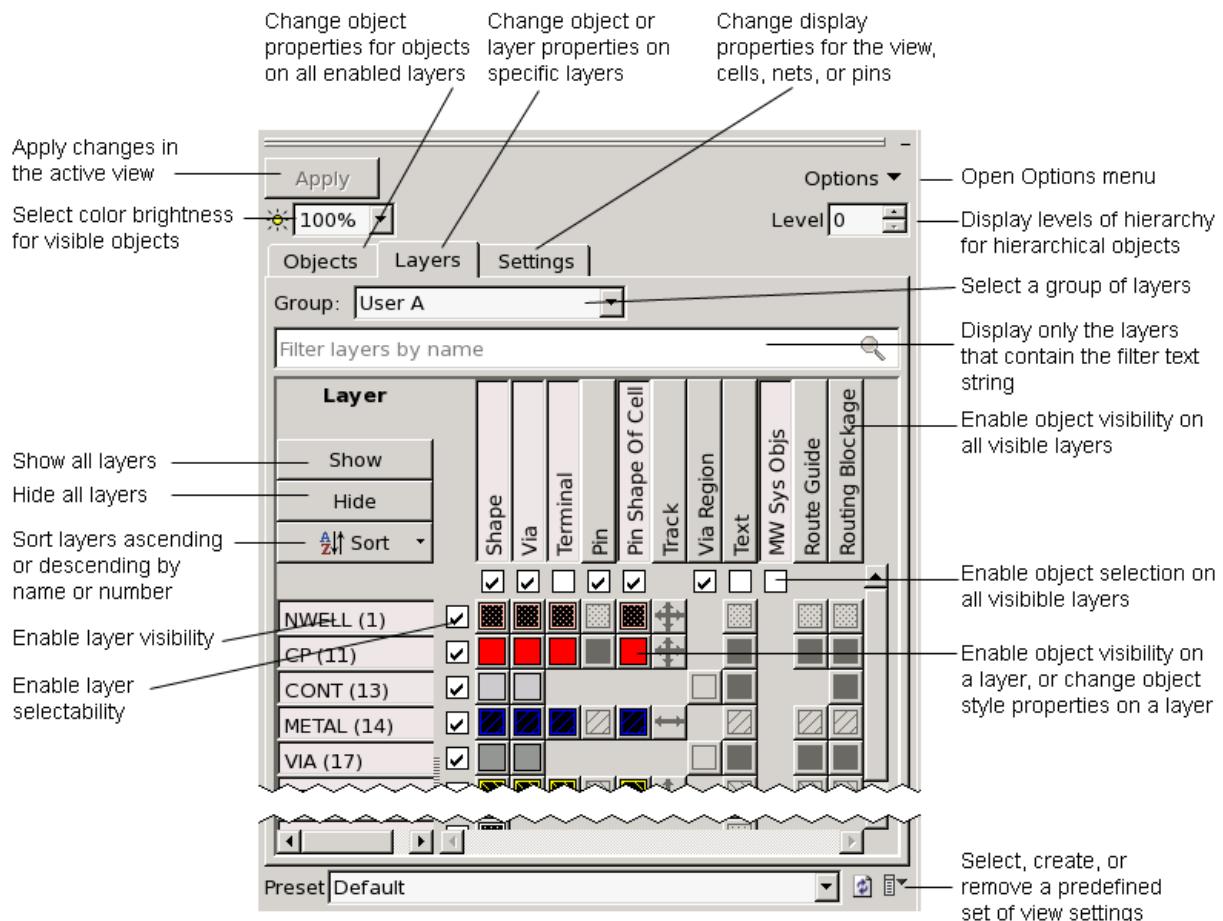
You can set options individually, or you can apply a predefined set of view settings called a *preset*. You can apply a system preset provided by IC Compiler or a custom preset that you create.

[Figure A-12](#) shows the View Settings panel as it appears when you open a new layout window, and explains what you can do with it.

Figure A-11 View Settings Panel for Objects in Layout Views



[Figure A-12](#) shows the View Settings panel as it appears when you click the Layers tab to set options for layers or objects on layers.

Figure A-12 View Settings Panel for Layers in Layout Views

To change layout view display properties in the active layout view,

1. Set options as needed on the View Settings panel.
2. Click **Apply**.

By default, when you change settings on the View Settings panel, you must click **Apply** before the changes take effect in the active view. If you prefer, you can set the panel to automatically apply your changes as soon as you make them.

To enable or disable the automatic apply mechanism,

- Choose **Options > Auto Apply**.

A check mark beside the command on the Options menu indicates that the auto apply mechanism is enabled.

Alternatively, when the automatic apply mechanism is not enabled, you can reverse changes that you have not already been applied.

To reverse unapplied changes,

- Choose Options > Cancel.

To filter the list of layers,

- Enter a text string in the layer text filter box.

As you type, the layer settings panel updates to show only layers that contain the text you entered. Only the list of layers is updated; the design shown in the layout window is not affected by layer filtering. Click the (X) to the right side of the text box to clear the filter.

To sort the list of layers,

- Click the sort button and choose a sort option to reorder the list of layers.

The tool updates the Layer panel and sorts the layers by name or layer number, in ascending or descending order, based on the option you choose.

You can set visibility or selection options or change style properties for

- Object types or subtypes
- All objects on a layer
- Object types on a layer

Object subtypes are categories of objects by property or attribute. For example, when cells are visible, you can display core cells and hide macro cells.

By displaying or hiding particular object types or layers, you can visually inspect just the physical layout data that you are interested in viewing while ignoring unrelated data. By enabling or disabling selection for particular object types or layers, you can control which types of objects are selected when you click or drag the pointer in a layout view.

You can customize how objects appear in the layout view by changing their style properties. Object styles set the appearance of objects in the active layout view. You can set the color, fill pattern, outline style, outline width, or exaggeration value for individual object types or layers.

Note:

The exact properties that you can change depend on an object's graphic representation in the layout view.

Typically you customize object style properties for a design. For example, the colors you use for individual layers might change, depending on the number of layers in the design.

The default display characteristics for design objects in a layout view are set for optimal viewing. These default characteristics work well for most designs. However, you can change

the display characteristics for individual object types if you need to customize your layout views for a particular design or working environment.

For more information about using the View Settings panel with the active layout view, see the “Configuring the Layout View” topic in IC Compiler Help.

Saving and Restoring Layout View Settings

IC Compiler does not save view settings when you exit the session. If you change layout view settings during a session and want to use the same settings in a future session, you can save them in your preferences file. You can also restore previously saved view settings by loading them from your preferences file.

To save the current settings for the active layout view,

- Choose Options > Preferences > Save to Preferences on the View Settings panel.

To restore the most recently saved layout view settings,

- Choose Options > Preferences > Set from Preferences on the View Settings panel.

For more information, see the “Setting and Saving View Properties” topic in IC Compiler Help.

Applying Predefined Layout View Settings

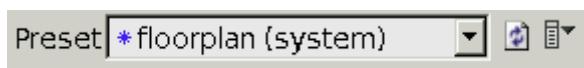
You can configure the layout view by applying a predefined set of view properties called a *preset*. You can apply a system preset provided by IC Compiler, and you can create and apply your own custom presets.

By default, when you open a layout window, the tool configures the layout view by using the view settings saved in your preferences file. To apply view settings from a preset, select the preset in the Preset list at the bottom of the View Settings panel.

For example, to apply the system preset that configures the layout view for design planning, select “floorplan (system)” in the Preset list.



When you change settings manually, by setting options on the View Settings panel, an asterisk sign (*) appears in front of the preset name in the Preset list.



If you need to reapply the view settings from the preset, click the  button. To reconfigure the layout view with the default view settings, select Default in the Preset list.

When you create a custom preset, you specify a preset name and the view settings that you want to include in the preset. You can create a new preset or redefine an existing custom preset. You cannot redefine a system preset.

To create a custom preset,

1. Click the  button and choose “Save settings as preset.”

The Write Layout Settings Preset dialog box appears.

2. Select an existing preset, or type a new preset name in the “Select or type a preset name” box.

3. Set options to control which settings you want to include.

The object display properties from the Objects tab and the layout display settings from the Settings tab are included by default. You can set options to include layer display properties from the Layers tab or object style properties from the Objects and Layers tabs.

4. Click OK.

The tool saves the preset and adds the preset name to the Preset list on the View Settings panel.

You can specify a preset as the default configuration when you open a new layout window or layout view. You can remove a custom preset but not a system preset.

For more information, see the “Selecting Preset Options” topic in IC Compiler Help.

Viewing Design Overlays

You can use the View Settings panel to overlay other designs on the primary design in the active layout view. For example, to view the fill data for a design, you can display the FILL view as an overlay over the CEL view. Similarly, you can view changes in a design by displaying an earlier version as an overlay over the current version.

To add an overlay on the active layout view,

1. On the View Settings panel, click the Settings tab and then the Overlays tab.

The panel lists the names of the current overlay designs and their brightness levels.

2. Click the Add button.

The Add Overlay Design dialog box appears, with a list of the designs that are open and available to be used as layout view overlays.

3. (Optional) If you need to open a design and display it as an overlay, click the Open Design button, select the design name in the Open Design dialog box, and click OK.

Note:

The GUI does not open a new layout window for a design you open to add as an overlay. However, when you open a new layout window (by choosing Window > New Layout Window), you can select any design as the primary design for the window.

4. Select the design name in the Add Overlay Design dialog box, and click OK.

The tool displays the design name on the Overlays tab, sets the brightness level to 50 percent, and displays the overlay on the active layout view.

When you add an overlay to the active layout view, the tool displays the design name on the View Settings panel and sets the brightness level to 50 percent. You can display or hide an overlay by controlling its brightness, but you cannot select, query, or edit objects in an overlay design.

To remove an overlay from the active layout view,

1. On the View Settings panel, click the Settings tab.

2. Click the Overlays tab.

3. Click the Remove button for the design you want to remove.

A message box appears asking if you want to close the design. Do one of the following:

- To keep the design open, deselect the “Close design after removing” option and click OK.
- To close the design, click OK.

If you have unsaved design changes, the Close Design dialog box appears. Set options to save or discard design changes as needed and click OK to close the design, or click Cancel to keep the design open without saving the changes.

For more information, see the “Viewing Design Overlays” topic in IC Compiler Help.

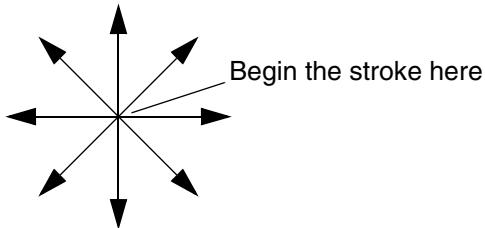
Using Stroke Activated Commands

You can stroke the pointer to activate preset commands, which can include icc_shell commands, in the active layout view. By default, the strokes are set up to activate commands that do the following:

- Increase or decrease the magnification of the design
- Pan the design
- Display the entire design

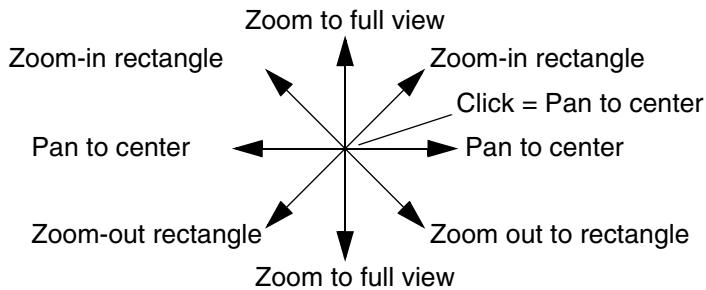
You can also define your own strokes. You can define up to 72 basic strokes and many more multisegmented strokes.

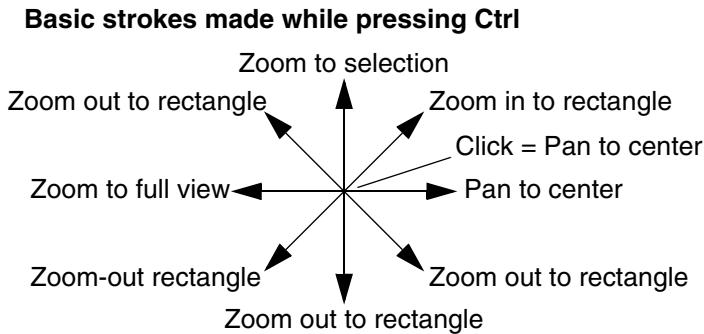
A basic stroke is a single click of the middle mouse button with the pointer in the layout view. A point-to-point stroke (dragging the pointer from point to point in one of the following directions while holding down the middle mouse button).



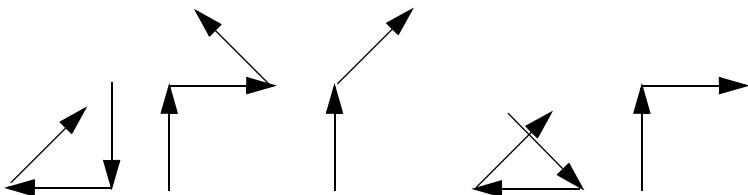
You can activate other commands for each of these strokes by also pressing a modifier key (Shift, Ctrl, or Alt), or a combination of these keys as shown in the following figures:

Basic strokes made without pressing Shift, Ctrl, or Alt





In addition to the basic and modified stroke commands, you can perform multisegmented strokes. Each segment can go in any one of the eight directions used for the basic stroke commands. The following drawing shows some of the simple multisegmented strokes you can define:



To bind a command to a basic or multisegmented stroke, use the `set_gui_stroke_binding` command. By default, the GUI is set up to use basic strokes instead of multisegmented strokes. You can change this preference by using the `set_gui_stroke_preferences` command.

Analyzing the Physical Design

The layout window provides tools and commands that you can use to visualize and analyze your design at each stage in the design process. These tools and commands are described in the following sections:

- [Using Visual Analysis Modes](#)
- [Tracing Paths Through the Design](#)
- [Analyzing the Data Flow for Macro Placement](#)
- [Examining and Editing Relative Placement Groups](#)

- [Analyzing Signal Integrity](#)
- [Analyzing Clock Trees](#)

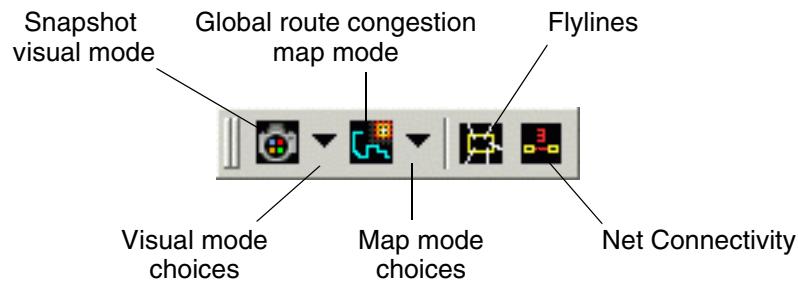
Using Visual Analysis Modes

The layout view provides visual analysis modes that help you analyze the physical design by coloring specific types of design or timing data. You can use the Analysis toolbar, shown in [Figure A-13](#), to enable or disable the following visual analysis modes in the active layout view:

- Flylines
- Net connectivity
- Map modes
- Visual modes

The Analysis toolbar is available only in the layout window.

Figure A-13 Analysis Toolbar



In addition, the layout window provides tools that you can use to examine relative placement groups, analyze signal integrity, and examine routing and verification errors.

You can use the GUI to analyze the physical layout as explained in the following sections:

- [Displaying Flylines](#)
- [Displaying Net Connectivity](#)
- [Using Visual Modes](#)
- [Using Map Modes](#)

Displaying Flylines

Flylines let you view the connectivity of selected objects (cells, pins, or ports) in the layout window. You can select an object and display flylines to see the locations of the objects that have net connections to the selected object.

Flylines represent unrouted straight-line pin-to-pin connections. By default, the layout window shows flyline connections from the selected objects to macro cells, I/O ports, and other selected objects.

Note:

Make sure the desired object types (core cells, macro cells, I/O cells, pins, or ports) are visible and can be selected in the layout view (select their visibility and selection options on the View Settings panel).

To view the flylines from selected cells,

1. In the layout window, click the  button on the Analysis toolbar or choose View > Flylines.

The Flyline Settings panel appears.

2. (Optional) Set options as needed on the Flyline Settings panel.

You can control how the flylines are displayed and change their appearance.

3. Select one or more cells, pins, or ports.

Flylines appear from the selected objects to the cells, pins, ports, or other selected objects with which they are connected. You can

- Zoom in if necessary to see individual flylines
- Pan or scroll to follow a flyline through the design

4. Select or deselect objects as needed to view their flylines.

If you highlight an object while its flylines are displayed, the flylines are also highlighted, and they remain highlighted when the object is no longer selected.

Note:

A flyline is a line that represents a single connection between two objects such as pins or cell instances. If you need to visualize the connectivity for an object such as a plan group that requires multiple connection lines, use the net connectivity display.

For more information, see the “Displaying Flylines” topic in IC Compiler Help.

Displaying Net Connectivity

Net connectivity lets you view the connections of selected nets in the layout window. You can select a net and display net connectivity to see the locations of the objects that have connections to the selected net.

Net connectivity displays the net connections to an object as a line with a number representing one or more nets between objects. These objects can be macro cells (including soft macros, hard macros, and black boxes), I/O cells, bounds, plan groups, or voltage areas. Net connections represent unrouted straight-line pin-to-pin connections. By default, the layout window shows net connections from a selected net to macro cells, I/O ports, and other selected objects.

Note:

Make sure the desired net types are visible and can be selected in the layout view (select their visibility and selection options on the View Settings panel).

To view the connections from selected nets,

1. In the layout window, click the  button on the Analysis toolbar or choose View > Net Connectivity.

The Connectivity Settings panel appears.

2. (Optional) Set options as needed on the Connectivity Settings panel.

You can control how the net connections are displayed and change their appearance.

3. Select one or more cells, pins, or ports.

Net connections appear from the selected objects to the cells, pins, ports, or other selected objects with which they are connected. You can

- Zoom in if necessary to see individual net connections
- Pan or scroll to follow a net connection through the design

4. Select or deselect objects as needed to view their net connections.

If you highlight an object while its net connections are displayed, the connections are also highlighted, and they remain highlighted when the object is no longer selected.

For more information, see the “Displaying Net Connections” topic in IC Compiler Help.

Using Visual Modes

Visual modes are analysis tools you can use to illuminate specific design or timing information with colors in the active layout view. When you activate a visual mode, the GUI dims the visible objects in the layout view and opens the Visual Mode panel. The name of the active visual mode appears in the list at the top of the Visual Mode panel.

For power network analysis in a multivoltage design, you can highlight voltage areas and level shifters.

For floorplan and placement analysis, you can use visual modes to highlight

- Block placement
- Cells by hierarchy
- Illegal cell locations
- Worst slack through cells
- Net connections in an area
- Relative placement groups
- Relative placement net connections
- Scan chains
- Categorized timing reports

For routing analysis, you can use visual modes to highlight

- Worst slack timing paths
- Timing path pins
- Net capacitance

For signal integrity analysis, you can use visual modes to highlight

- Crosstalk victim and aggressor nets
- Static noise
- Switching noise

For clock tree analysis, you can use visual modes to highlight

- Clock trees
- Clock tree timing (latency or transition)

A visual mode groups design objects, timing data, or other information into categories or ranges called bins. The layout view displays each bin in a different color.

- To enable or disable visual mode, click the visible visual mode button on the Analysis toolbar or choose View > Visual Mode.

Clicking the visual mode button activates the default visual mode (snapshot) or the most recently active visual mode. The button icon changes to indicate the active visual mode.

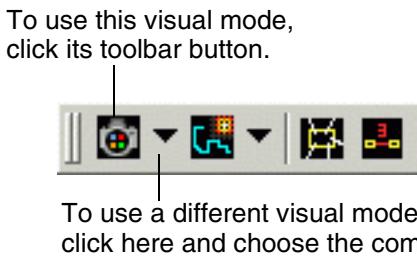
- To activate a particular visual mode, choose a command from the Visual Mode menu on the Analysis toolbar.
- To change to a different visual mode, select its name in the list on the Visual Mode panel.

Note:

The first time you activate some visual modes during a session, you must load the data before you can view the color overlay in the layout view.

[Figure A-14](#) shows the default visual mode button and the Visual Mode menu button.

Figure A-14 Visual Mode Tools on the Analysis Toolbar



Each visual mode displays a legend of colored bars on the Visual Mode panel. The legend colors correspond to colors in the overlay. You can click the visibility options at the left end of each bar to display or hide individual bins in the layout view.

From left to right, the legend consists of

- A column of visibility options
- A column of labels (names or range values) that identify the bins
- Columns of information about the bins, including (by default)
 - Color and fill pattern (colored boxes)
 - Count (total items in a bin)
 - Exaggeration value in pixels (hidden by default)
- A histogram showing the distribution of objects or values by the relative lengths of the colored bars

You can change the color styles for individual bins, hide or display information columns on the legend, and adjust the color exaggeration.

In visual modes that color design objects, you can select or deselect the objects in each bin. In visual modes that color discrete, unrelated sets of objects or other information, you can reorder the bars in the legend.

Using Map Modes

Map modes are analysis tools you can use to illuminate specific floorplan, placement, or routing results with colors in the active layout view. When you activate a map mode, the GUI dims the visible objects in the layout view and opens the Map Mode panel. The name of the active map mode appears in the list at the top of the Map Mode panel.

You can generate color maps to display

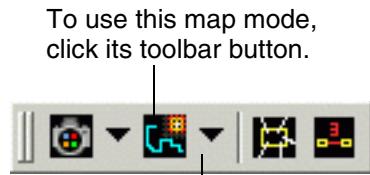
- Congestion (global route, track assignment, or detail route)
- Preroute power network analysis (electromigration, voltage drop, or resistance)
- Cell placement (cell density or pin density)
- Signal electromigration
- Critical areas (shorts and opens)
- Static postroute In-Design rail analysis (electromigration, voltage drop, resistivity, parasitics, or power)
- Dynamic postroute In-Design rail analysis (electromigration or voltage drop)

A map mode divides the core area into an array of boxes with colored edges. The box edges are colored and labeled to represent the map data (such as congestion or power density) through the horizontal and vertical planes. Each map color represents a range of values.

- To enable or disable map mode, click the map mode button on the Analysis toolbar or choose View > Map Mode.
Clicking the map mode button activates the default map mode (global route congestion) or the most recently active map mode. The button icon changes to indicate the active map mode.
- To display a particular map mode, choose a command from the Map Mode menu on the Analysis toolbar.
- To change to a different map mode, select its name from the list on the Map Mode panel.

[Figure A-15](#) shows the default map mode button and the Map Mode menu button.

Figure A-15 Map Mode Tools on the Analysis Toolbar



To use a different map mode, click here and choose the command.

Each map mode displays a legend of colored bars on the Map Mode panel. The legend colors correspond to colors in the map. You can click the visibility options at the left end of each bar to display or hide individual bins in the layout view.

From left to right, the legend consists of

- A column of visibility options
- A column of labels (names or range values) that identify the bins
- Columns of information about the bins, including (by default)
 - Color and fill pattern (colored boxes)
 - Count (total items in a bin)
 - Exaggeration value in pixels (hidden by default)
- A histogram showing the distribution of map data by the relative lengths of the colored bars

You can change the color styles for individual bins, hide or display information columns on the legend, and adjust the color exaggeration.

Tracing Paths Through the Design

You can trace objects on a path through the design and visually examine the path in the layout view. You define the path by tracing nets in the Netlist Tracer dialog box. The tool traces nets between cells or between a cell and a port. As you trace the path, you can view and highlight the cells, pins, ports, and nets in the layout view.

You begin tracing a path by specifying its startpoint. The Tracing Objects browser in the Netlist Tracer dialog box displays the names of the cells and nets along the path. You can expand a cell or net to display its associated pins and ports. You can go back to any cell or net on the path and resume tracing the path in a different direction.

To begin tracing a path,

1. Choose Highlight > Netlist Tracer in the layout window.

The Netlist Tracer dialog box appears.

2. Select the startpoint object and click the button.

You can select the object before or after you open the Netlist Tracer dialog box.

Alternatively, you can select the object type in the Type list and type or select the object name in the box beside the Type list. You also can begin typing the object name, press Tab to display a list of matching names, and select a name in the list.

3. Click the Trace button or press the Return key.

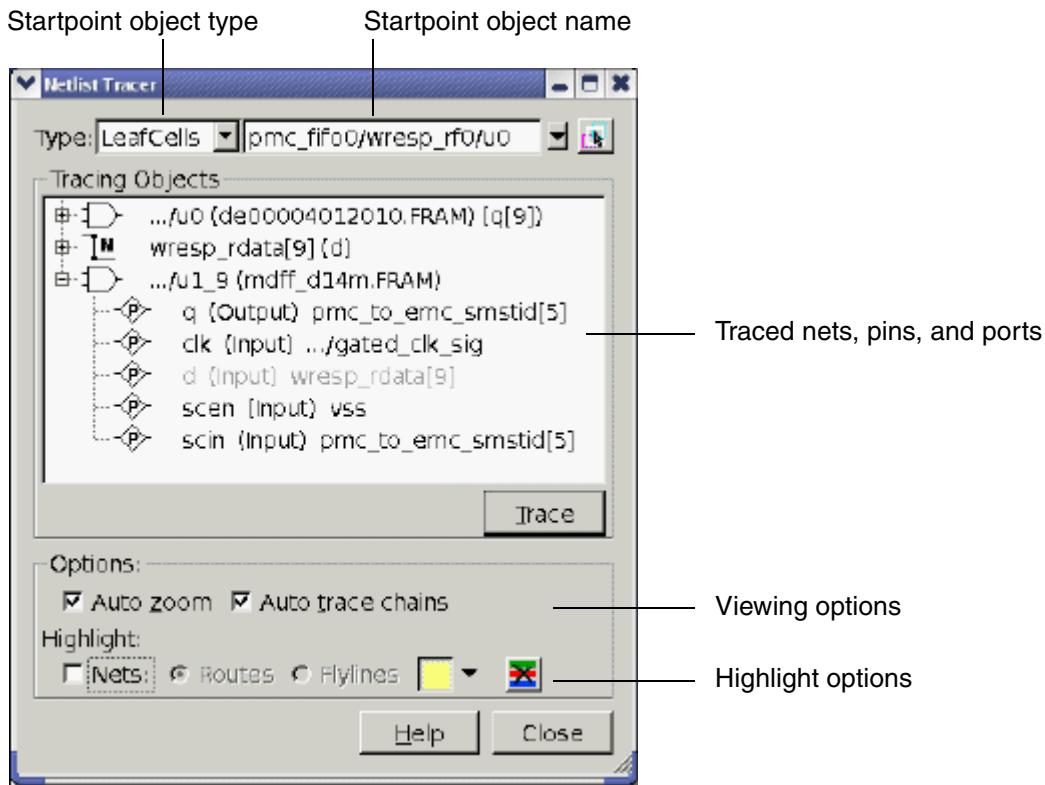
The Tracing Objects browser displays a history of the cells and nets on the path that you have traced. The first row shows the startpoint. The last row shows the most recent cell or net that you have explored.

You can expand a cell row to display the input and output pins on the cell or expand a net row to display the input and output pin or port connections on the net. The last row is expanded by default. The row color changes to light gray for already-visited pins and ports.

To continue tracing objects on the path,

1. Select a pin or port in the Tracing Objects browser.
2. Click the Trace button.

[Figure A-16](#) shows an example of the Netlist Tracer dialog box after the path has been traced from the startpoint cell to another cell.

Figure A-16 Netlist Tracer Dialog Box

You can select objects in the Tracing Objects browser to view them in the layout view.

- When you select a cell, pin, or port, the tool automatically selects and highlights it in the layout view.

By default, the tool magnifies the view to fit the selected object. You can disable this mechanism by deselecting the “Auto zoom” option.

- When you select a net, the tool displays flylines in the layout view showing the net connections from the most recently examined pin or port.

To highlight nets in the layout view, select the Nets option. Select the Flylines option if you want to display the nets as flylines instead of routed nets.

The tool highlights the selected objects with the current highlight color, which is yellow by default. You can change the highlight color or enable the automatic color cycling mechanism. You can also clear the colors from all highlighted objects.

For more information, see the “Tracing Paths Through the Design” topic in IC Compiler Help.

Analyzing the Data Flow for Macro Placement

The GUI provides tools that you can use to analyze and improve the data flow through the physical design. These tools provide information about the data flow in a way that can help you to decide what to do next to improve the placement of hard macros in your design.

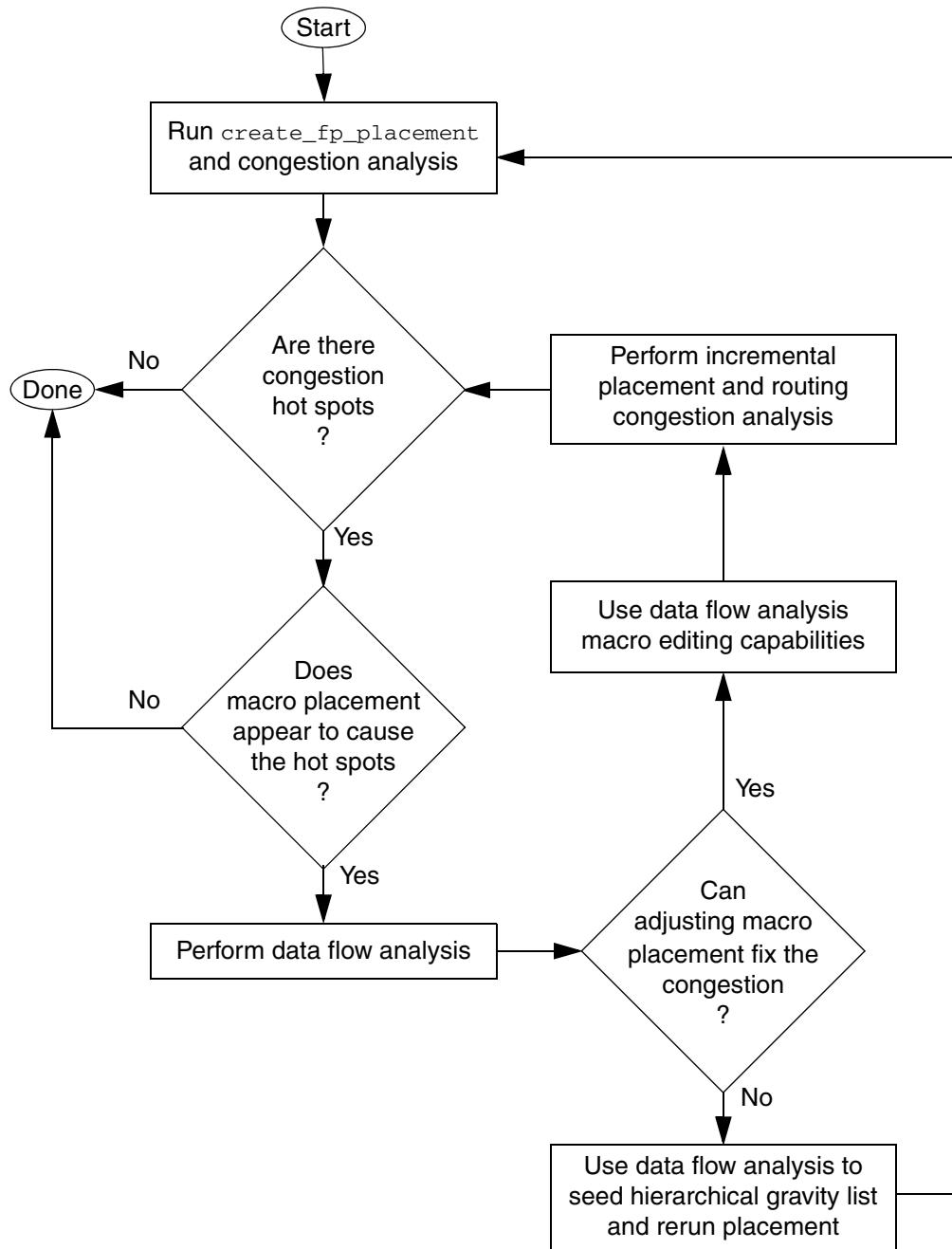
You can use tools in the data flow analysis window to perform the following tasks:

- Logic connectivity analysis allows you to view macro connections from the perspective of the logic data flow. You can limit the analysis by net fanout, register levels, gate levels, and number of flylines.
- Advanced flyline analysis allows you to identify the data flow by examining connectivity between macros and between macros and I/O cells.
- Interactive macro editing allows you to modify macro placement, create and customize macro relative placement groups, create macro arrays by dragging and dropping macros into array locations, and copy macro array placement layouts.

After performing your initial macro placement in IC Compiler, you should analyze your floorplan to find areas of high congestion. If your macro placement appears to be causing congested hot spots, you can use the data flow analysis tools to examine hard macro placement and connectivity and to edit the macro placement.

[Figure A-17](#) shows when and how to use data flow analysis in the macro placement flow. Data flow analysis augments the automatic macro placement capabilities of the `create_fp_placement` command and provides analysis and editing tools that can help you to adjust macro placement manually.

Figure A-17 Data Flow Analysis in the Macro Placement Flow



Data flow analysis is not intended to replace automatic macro placement.

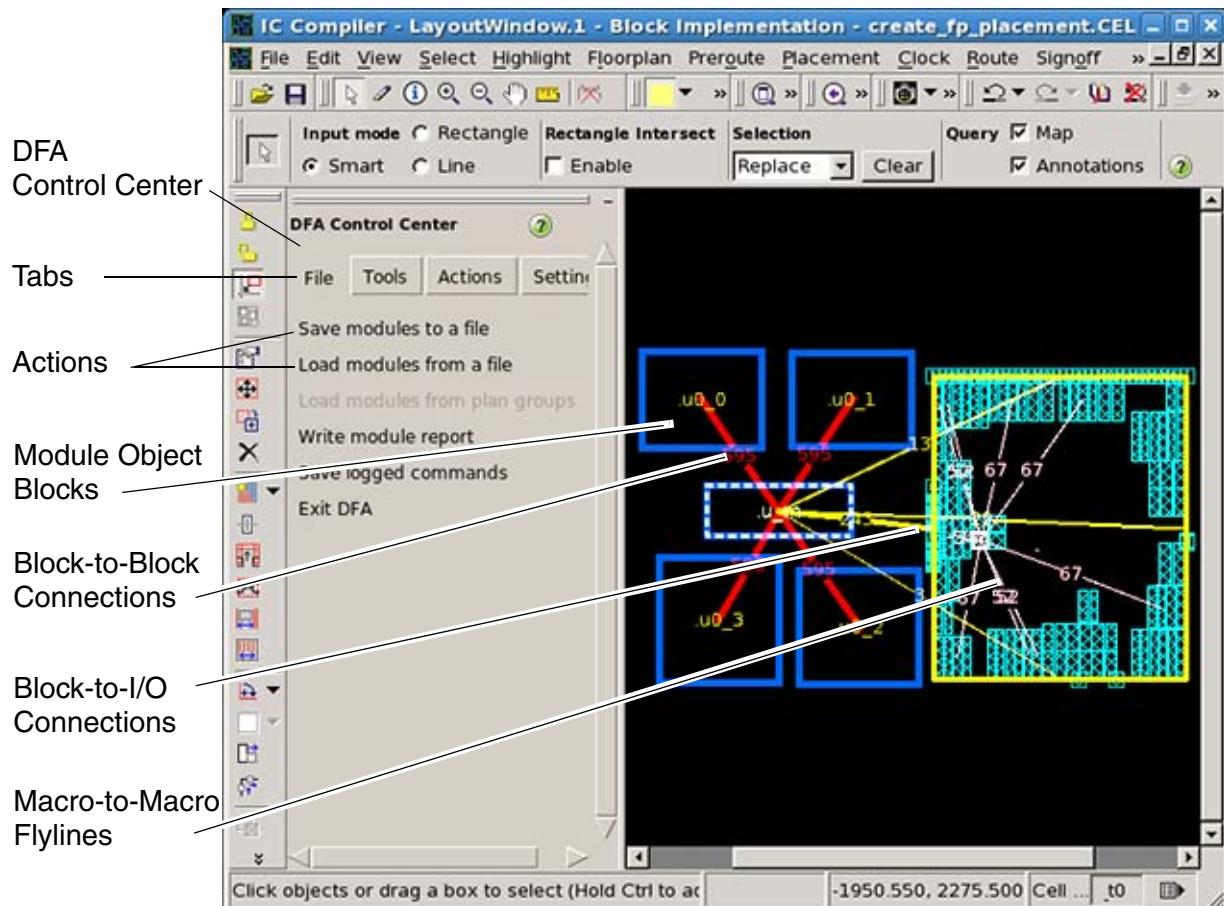
To open the data flow analyzer window,

- Choose Floorplan > Analyze Logic Connectivity.

Alternatively, you can use the `analyze_logic_connectivity` command.

The DFA Control Center panel opens and the layout updates to show the module object blocks for the design. [Figure A-18](#) shows the layout after selecting and expanding the top-level module object block.

Figure A-18 Data Flow Analysis View



For information about the operation of the data flow analysis tool, see Appendix B, “Data Flow Analyzer,” in the *IC Compiler Design Planning User Guide*.

Examining and Editing Relative Placement Groups

A relative placement group can contain leaf cells, hierarchy (other groups), and placement keepouts. A hierarchy level is a relative placement group embedded in another relative placement group, either in the same design (included groups) or in a different design or designs (instantiated groups).

The IC Compiler GUI provides tools you can use to interactively examine and edit relative placement groups.

In the layout window, you can

- Explore relative placement group structures by using the relative placement hierarchy browser
- Examine relative placement groups by using the relative placement visual mode
- Examine relative placement group net connections by using the relative placement net connections visual mode
- Move relative placement groups in the layout view by using the Move/Resize tool

In the relative placement window, you can

- Explore relative placement group structures by using the relative placement hierarchy browser
- Visually examine the objects in a relative placement group by using interactive mouse tools in a relative placement view

You can select, highlight, and query objects in the group, and you can select rows or columns in the group.

- Edit a relative placement group by using the interactive editing tools in a relative placement view

You can move switch, add, and remove objects in the group, and you can add, remove, and flip rows or columns in the group.

For more information, see “[Working With Relative Placement Groups in the GUI](#)” on [page 9-69](#) and the “Examining and Editing Relative Placement Groups” topic in IC Compiler Help.

Analyzing Signal Integrity

The GUI provides tools you can use to analyze the affects of crosstalk by examining the victim and aggressor nets. The two major effects of crosstalk are crosstalk-induced delay and static noise.

- Crosstalk-induced delay occurs when the victim and aggressor nets switch at the same time or nearly the same times, either speeding up or slowing down the victim net transition times.
- Static noise is a glitch on the victim net that occurs when the victim net is in a steady state of either a high or low, and the aggressor net is switching.

You can analyze crosstalk problems on a design after performing global routing, track assignment, or detail routing. The minimum requirement is a global-routed design. You can

- Examine a high-level view of the delta delays for victim nets in the routed design by using the delta delay visual mode
- Examine a high-level view of the accumulated static noise levels for victim nets in the routed design by using the static noise visual mode
- View information about individual victim nets and their aggressor nets in the static noise analysis window or the path inspector window
- Highlight a victim net and its aggressor nets in the layout view by using the crosstalk visual mode
- Check the current flowing through metal wires and vias by using the signal electromigration map mode

Before analyzing crosstalk to find static noise and crosstalk-induced delay problems, you can set crosstalk-related timing setup options and specify supply voltages. You can set these options by using the Set SI Options dialog box (choose Timing > Set SI Options) or the `set_si_options` command.

For more information about performing crosstalk analysis, see “[Analyzing Crosstalk](#) on page 8-7” and the “Analyzing Signal Integrity” and “Inspecting Victim and Aggressor nets” topics in IC Compiler Help.

Analyzing Clock Trees

You can use tools in the layout window and the interactive clock tree synthesis (CTS) window to perform both high-level and detailed analyses of the clock tree structures and timing in your design.

- In the layout window, you can use visual modes to view the overall structures or timing distribution of the synthesized clock trees.
- In the interactive clock tree synthesis window, you can view information about all the clocks in the design and detailed information about individual clock trees.

The interactive clock tree synthesis window provides tools you can use to

- View the distribution of clock tree pin arrival times in a clock tree arrival histogram
Clock arrival histograms provide a high-level overview of the timing quality in the fanout network of a selected clock tree.
- View clock details and select clocks on the Clock Browser panel

- Explore the pin-to-pin fanout structure of a clock tree in the clock tree fanout browser
The Clock Browser panel and the clock tree fanout browser are the starting points for detailed clock tree analysis. You can open one clock tree fanout browser for each clock tree in the design.
- Examine the fanin or fanout network for an entire clock tree or selected clock tree structures in clock tree network schematics
- Examine clock tree latency over time for individual clock trees in latency abstract clock graph views
- Examine the fanout structure of clock networks before and after clock tree synthesis in leveled abstract clock graph views
- Visualize the relationships between driving cells and output loads in a clock network or the relationships between clock trees in abstract clock graph views
- Find, select, and view information about specific clock tree objects (cells, pins, or load nets) in clock tree synthesis object list views

You can highlight the maximum or minimum fanout path through a clock tree in the clock tree fanout browser or a clock tree schematic.

To open an interactive clock tree synthesis window,

- Choose Window > New Interactive CTS Window.

The Clock Browser panel at the top of the window displays a list of the clocks in your design with information about each clock. You can use the Clock Browser panel to generate a clock arrival histogram or display the selected clock tree in a clock tree fanout browser.

You can select clock tree objects in the interactive clock tree synthesis window to highlight them in the layout view or color them in the clock trees visual mode. For information about layout view visual modes, see “[Using Visual Modes](#)” on page A-72.

In both the layout and interactive clock tree synthesis windows, you can use clock tree synthesis menu commands to perform tasks such as viewing clock tree options and setting, changing, or removing clock tree options, exceptions, and references.

For more information about analyzing clock tree structures and timing, see “[Analyzing Clock Trees in the GUI](#)” on page 5-143 and the “Analyzing Clock Tree Synthesis” topic in IC Compiler Help.

Analyzing Timing Paths

The timing analysis window provides a centralized area for performing timing path analysis. The window includes a timing analysis driver, a path inspector, and other analysis tools such as histograms, schematics, path profiles, and timing reports.

You can use tools in the timing analysis window to perform both high-level and detailed analyses of the timing paths in your design. You can

- Examine timing path details in the timing analysis driver
- View histograms that show the distribution of worst path slack, endpoint slack, net capacitance, or other path details in histogram views
- Inspect the clock network and path data elements for a selected path in the path inspector
- View path delay profiles for selected paths in path profile views
- View selected paths, fanin and fanout cones, and other objects in path schematics
- View and save timing reports for selected paths or the paths with the worst slack in the design

The following sections describe the timing analysis tools:

- [Opening a Timing Analysis Window](#)
- [Viewing Timing Path Details](#)
- [Using Timing Analysis Views](#)
- [Viewing Path and Design Schematics](#)
- [Inspecting a Timing Path](#)

Opening a Timing Analysis Window

When you open a timing analysis window, you can set the criteria for displaying timing path details in the timing analysis driver. By default, details for the 20 worst paths in each path group are displayed.

To open a timing analysis window,

1. Choose Window > New Timing Analysis Window.
The Select Paths dialog box appears.
2. (Optional) To display path details in the window, select a method for choosing the paths, and select other options as needed.

3. Click OK or Apply.

Note:

You can open the timing analysis window without displaying path details by clicking Cancel in the Select Paths dialog box when it appears.

Viewing Timing Path Details

When you open a timing analysis window, the timing analysis driver panel appears, attached to the left side of the window. The timing analysis driver displays detailed information about the timing paths that have the worst slack in the design. You can view the timing path details and select paths for further examination with other analysis tools.

The timing analysis driver consists of a timing path table, a button bar, and a command display box.

- The table displays a list of paths found by the tool, based on criteria that you specify. By default, the tool uses the `get_timing_paths` command.
- The button bar provides commands you can use to reload the paths, customize the table columns, save the path details in a text file, and access other timing analysis tools.
- The command display box shows the command and options used to find the paths.

The timing analysis driver is the focal point for timing path analysis in IC Compiler. You can select paths that you want to examine in another view, such as a schematic view, timing report, or path inspector window. You can also generate timing histograms based on the values in a column of the timing path table.

The first time you open the timing analysis driver during a session, the Select Paths dialog box appears. Use this dialog box to load timing path information in the timing path table. You can

- Set dialog box options for the `get_timing_paths` command.

The dialog box options are set by default to run the `get_timing_paths` command with its default options. You can reset the dialog box options to their defaults by clicking the Default button.

- Select a predefined collection command.

You can load all selected paths or all highlighted paths.

- Enter a command to define a custom collection.

The tool automatically adds the command to the list of predefined commands.

The tool displays the path names and other path information in the timing paths table and displays the command you used to load the paths in the command display box below the

button bar. You can copy text in the command display box and paste it somewhere else, such as on the console command line.

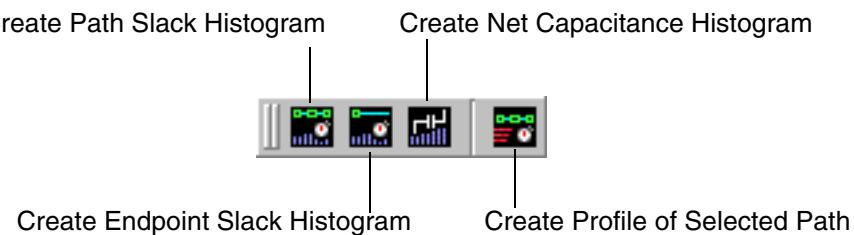
You can sort, filter, and customize the table, and you can save the path details in a text file.

For more information, see the “Examining Timing Path Details” topic in IC Compiler Help.

Using Timing Analysis Views

The Timing Views toolbar, which is shown in [Figure A-19](#), lets you generate histograms (path slack, endpoint slack, and net capacitance) and path profiles for timing analysis. The histogram buttons open dialog boxes in which you can set options to customize the histogram. The path profile button is enabled only when a timing path is selected. This toolbar can be found in the timing analysis window.

Figure A-19 Timing Analysis Toolbar



For more information about using timing analysis views, see the following sections:

- [Viewing Histograms](#)
- [Examining Path Profiles](#)

Viewing Histograms

You use histograms to view distributions of timing values such as slack or net capacitance. Histograms are focal points of visual timing analysis that allow you to view the overall timing performance of your design.

The IC Compiler GUI provides the following types of histograms for timing analysis:

- Endpoint slack histograms provide a high-level overview of the timing quality of your design. Create an endpoint slack histogram to identify endpoints that failed to meet their constraints.
- Path slack histograms provide a high-level overview of the timing quality for selected paths in your design. Create a path slack histogram to identify paths that failed to meet their constraints.

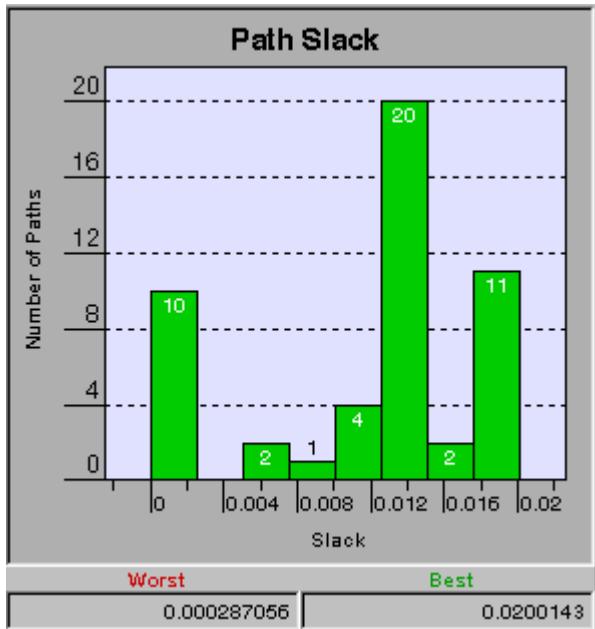
- Net capacitance histograms provide a high-level overview of the capacitance values for selected nets or for all the nets in your design. Create a net capacitance histogram to identify nets that have unacceptably high capacitance values.

In addition, when you use the timing analysis driver, you can create histograms that show the distribution of values for various types of path details, such as slack, endpoint clock skew, arrival time, or required time, that are listed in the timing path table.

In the interactive clock tree synthesis window, you can create a clock arrival histogram to identify clock network pins that failed their constraints. For details, see [“Viewing the Distribution of Clock Arrival Times” on page 5-148](#).

Histogram view windows are split into two panes. By default, the histogram bar graph appears in the left pane and an object table appears in the right pane. [Figure A-20](#) shows an example of the bar graph in a path slack histogram. The bins represent the number of paths (y-axis) versus their slack values (x-axis).

Figure A-20 Example of Path Slack Histogram



The number at the top of the tallest bin indicates the number of paths in the bin. Green bins (on the positive side of 0) contain paths that met their constraints. Red bins (on the negative side of 0) contain paths that failed to meet their constraints. You can

- Hold the pointer over a bin in the bar graph to display its value range and number of objects or paths in an InfoTip below the pointer
- Click a bin to display its contents (object names and slack or capacitance values) in the object table

- Select objects in the object table for further examination with other analysis tools such as path schematics or reports
- Filter the object table, limiting it to certain objects based on a character string or regular expression that you define

Examining Path Profiles

You can use the path profile view to visualize the relative importance of different cells and nets in contributing to the total delay of a timing path.

The path delay profile shows the pin-by-pin distribution of delays on a path—the same information that the timing report provides, except in graphic form. You can view two different types of profiles: hierarchical and flat.

The path profile view is identical to the path profile tool that you can view by clicking the Path Delay tab in a path inspector window.

To view the delay distribution along a timing path,

1. Select one or more timing paths.
2. Click the  button on the Timing Views toolbar or choose Timing > Path Profile View.
The path profile view window appears. The hierarchical profile appears by default the first time you open this view.
3. Click the tab for the type of path profile you want to view.
 - To view a hierarchical profile, click the Hierarchy tab.
 - To view a flat, leaf-level profile, click the Flat tab.

The combined delays for each path and the relative delay contribution for each pin appear graphically in bar graphs that represent the percentages of the total path delay. For each timing path, the bar graph shows the percentages of the individual pin delays from startpoint to endpoint. For each pin on a path, the bar graph shows the percentage contribution of the pin delay to the total path delay.

Viewing Path and Design Schematics

You can use the following schematic and symbol views to show graphical representations of the logic design:

- Path schematic view

Path schematic views show graphical representations of the logic design. You use these views to visually analyze timing and logic in the optimized design and to gather

information that can help you guide later optimizations. Objects you select in a schematic or symbol view are cross-selected in other views. This capability allows you to efficiently analyze the logic and timing aspects of your design.

- Design schematic view

Design schematics display a design as a schematic composed of cell instances, pins, nets, and ports. The schematic shows both leaf-level logic (gates) and subdesigns (blocks). You can move up or down in the design hierarchy to view the schematic for each block in the hierarchy.

- Symbol view

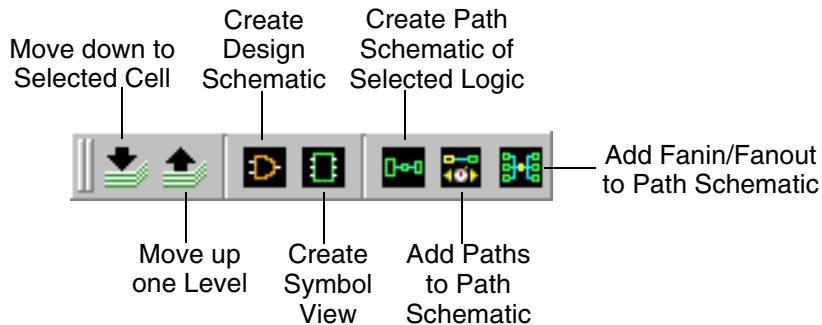
Symbol views display a hierarchical cell or a leaf cell as a black box with input and output ports. To change the view to a design schematic of the cell, double-click inside the box.

You use schematic and symbol views to visually analyze timing and logic in the optimized design and to gather information that can help you guide later optimizations.

Objects you select in a schematic or symbol view are cross-selected in other views. This capability allows you to efficiently analyze the logic and timing aspects of your design.

The Schematics toolbar, shown in [Figure A-21](#), lets you open design schematics (for selected designs), symbol views (for selected cells), and path schematics (for selected timing paths or design logic). When a design schematic is the active view, you can use this toolbar to move down or up a level in the design hierarchy. When a path schematic is the active view, you can use this toolbar to add timing paths, fanin logic, and fanout logic.

Figure A-21 Schematics Toolbar



You can use the View Settings panel to change the following display characteristics for the active schematic view:

- Object name visibility, text colors, and text sizes
- Object annotation visibility, text colors, and text sizes
- Object colors

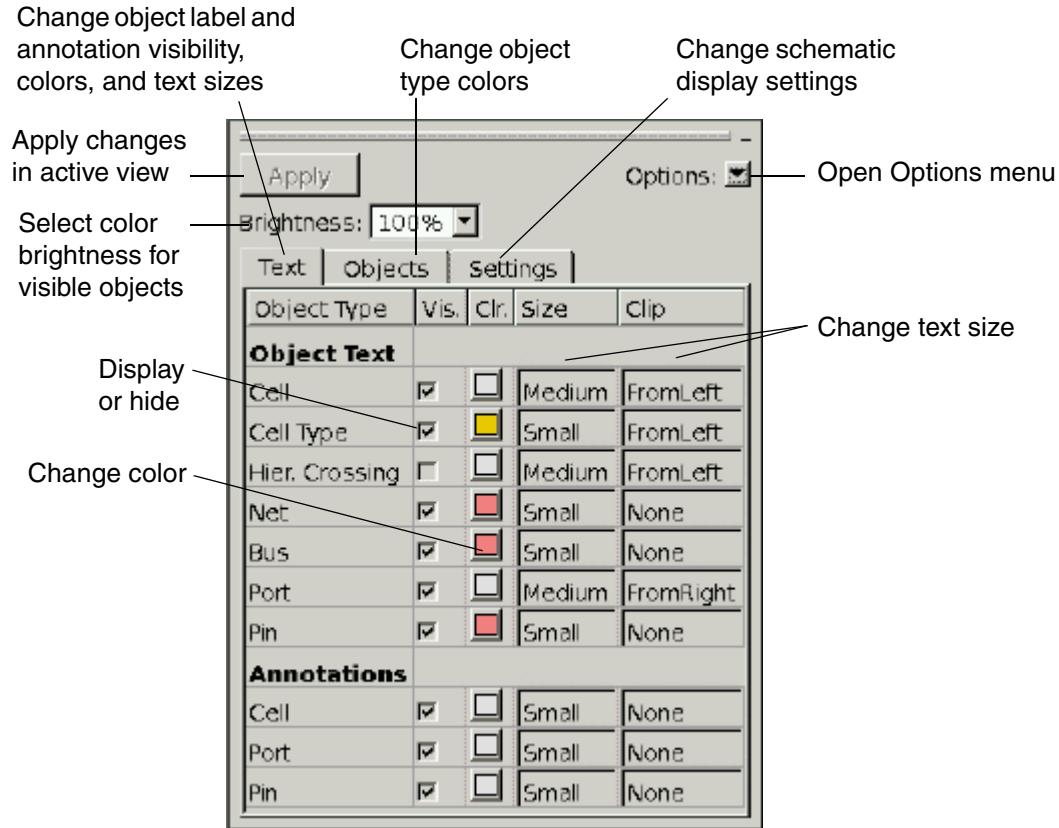
- Display style for highlighted timing paths

To display the View Settings panel,

- Choose View > Toolbars > View Settings.

[Figure A-22](#) shows the View Settings panel as it appears when a schematic view is the active view.

Figure A-22 View Settings Panel for Schematic Views



For more information about using schematic and symbol views, see the “Viewing Path and Design Schematics” topic in IC Compiler Help.

Inspecting a Timing Path

You can select an individual timing path and use the path inspector window to examine detailed information about the clock network and datapath elements in the path. The path inspector provides tools for viewing different aspects of the path, including

- The path status
- Path and clock summaries
- Clock network and path data elements
- Path delay profiles
- Crosstalk victim and aggressor nets
- A path schematic with clock path highlighting

To open a path inspector window,

1. Select the path you want to inspect.
2. Click the Inspector button on the timing analysis driver button bar.

Alternatively, you can choose **View > Show Path Inspector** or choose **Path Inspector > Show Path Inspector**.

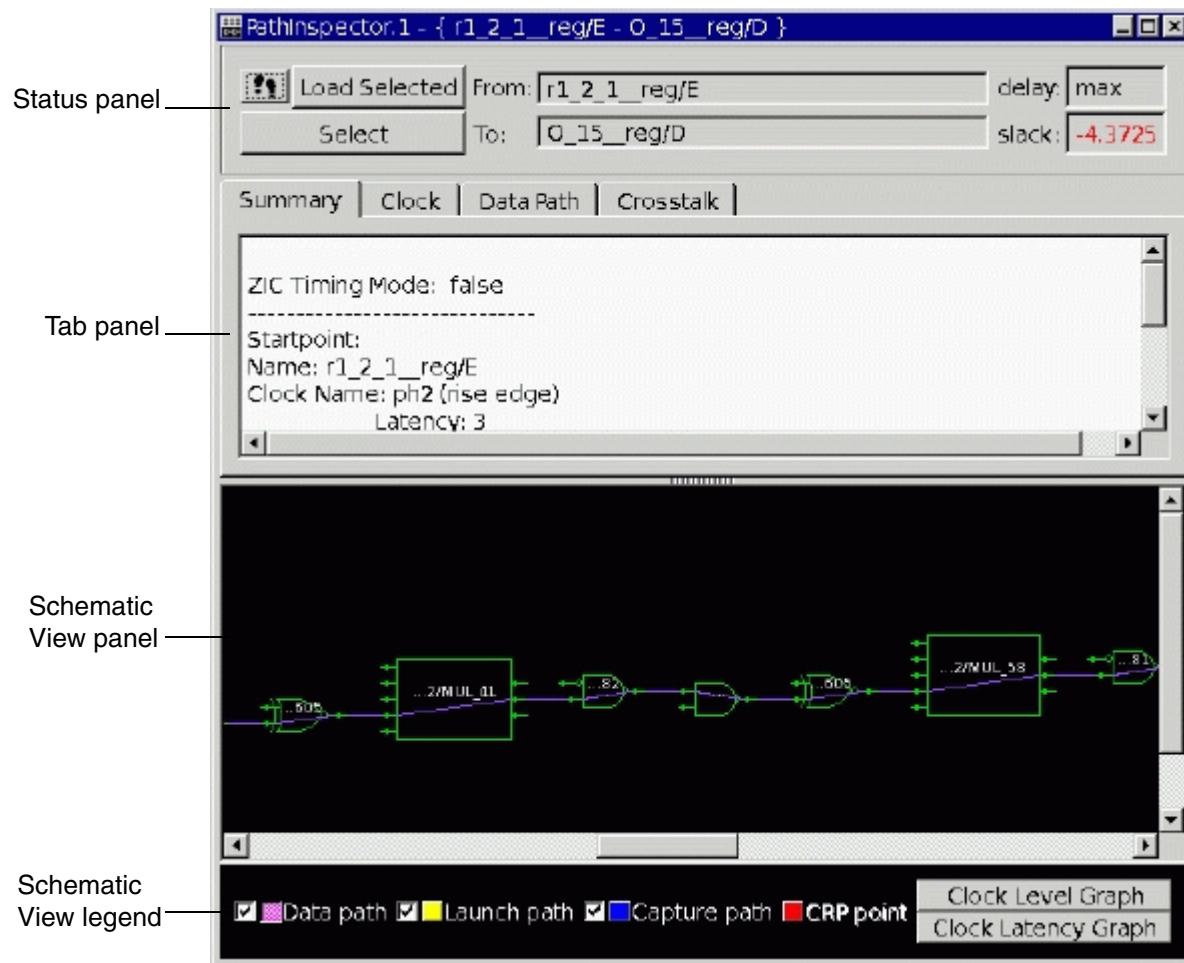
The path inspector window displays path information on three panels:

- The Status panel displays the path startpoint, endpoint, delay type, and slack. It also provides controls for automatically or manually loading another path.
- The Tab panel displays path summaries, clock network and path data element tables, path delay profiles, and crosstalk victim and aggressor net lists in a set of tabbed pages. You can copy the summaries and paste them into text files, export the tables to text files, customize the element tables, and generate delay calculation reports for selected cells or nets.
- The Schematic View panel displays a path schematic of the complete path (datapath, clock launch path, and clock capture path) with data and clock path overlays.

You can add fanin and fanout logic to the path schematic.

Figure A-23 shows an example of a path inspector window.

Figure A-23 Path Inspector View Window



The legend below the Schematic View panel provides options you can use to display or hide the paths and buttons you can use to display or hide the overlays.

If the timing path contains launch and capture clock paths, you can examine the clock paths in clock graph views. To examine driver and load relationships in a clock tree leveled-fanout graph, click the Clock Level Graph button. To examine clock skew, bottlenecks, insertion delays, and the intrinsic path delays by viewing a clock tree latency graph, click the Clock Latency Graph button. For details about using clock graph views, see “[Analyzing Clock Trees in the GUI](#)” on page 5-143.

For more information about using the path inspector, see the “[Inspecting Timing Paths](#)” topic in IC Compiler Help.

Examining Routing and Verification Errors

The error browser lets you examine routing design rule errors in your design. You can examine detail routing errors, routing and verification errors saved in error views, and editor DRC errors found during interactive route editing operations. You can organize the errors by error type or layer, view and sort the error list, view detailed error information in the error browser, and view error locations in the layout view. In addition, you can

- Filter the error data to display or hide errors based on attribute values, net associations, or both
- Mark errors as fixed or remove the fixed marker, optionally hide fixed errors, and save the updated error status
- Highlight errors in the layout view
- Display error information on the Query panel and optionally record the information in the session log
- Select errors interactively by using the Error Selection tool
- Save the error information in a report file
- Save error updates in an error view

You can also highlight the nets associated with detail routing errors and display the net nodes associated with open locator errors.

IC Compiler provides several routing and verification processes that check your design for rule violations and save the error data in files called error views. You can examine detail routing errors, layout versus schematic (LVS) verification errors, DRC violations, routability check violations, signoff DRC violations, In-Design rail analysis postroute power and ground verification errors, signal electromigration constraint violations, and third-party DRC violations. For more information about using these rule checking processes, see “[Analyzing the Routing Results](#)” on page 6-113.

You can also examine editor DRC errors that result from interactive editing operations in the layout view. The error browser loads editor DRC error view automatically when the tool detects rule violations, and editor DRC incrementally updates the error view when you fix a violation, reverse the edit that caused the violation, disable the rule used to detect the violation, or remove errors in the error browser. For more details about the editor DRC error view, see “[Examining Editor DRC Errors](#)” on page A-132.

Error views are associated with the current design. The error browser loads detail routing errors from memory and other types of errors from the appropriate error views. When you open an error view or load detail routing errors, the error browser can group the errors according to either the error types (the default) or the layers associated with the errors. The available error types depend on the verification process used to generate the error records.

Each error record contains information that includes a description string and bounding box coordinates correlated with the physical design. An error record can also provide references to design objects. This is the data that the error browser displays.

To open the error browser window,

- Choose Verification > Error Browser or Rail > Error Browser.

Note:

The Verification menu is available only in the block implementation task mode. To display the block implementation menus in the layout window, choose File > Task > Block Implementation or File > Task > All Tasks.

If you have already loaded detail routing errors or opened an error view during the session, the error browser window appears. For help using the error browser, choose Help > Error Browser in the error browser window.

If you have not already loaded detail routing errors or opened an error view, the Open Error View dialog box appears.

1. Select the options for the types of errors you want to examine, and deselect the options for the types of errors you do not want to examine.
2. For each option you select except Detail Route, type the error view name in the text box, or click the browse button and select the file in the Choose Cell dialog box.
3. Click OK.

When the error browser is open, if you need to reload an error view or load a different error view, choose File > Open in the error browser window to open the Open Error View dialog box.

For more information about examining errors with the error browser, see the following sections:

- [Viewing and Fixing Errors](#)
- [Filtering Errors in the Error Browser](#)
- [Selecting Errors in the Layout View](#)
- [Examining Nets Associated With Detail Routing Errors](#)
- [Examining Net Nodes Associated With Open Locator Errors](#)
- [Examining Errors in an Area](#)
- [Saving the Error List in an Error Report File](#)
- [Saving Error Updates in an Error View](#)
- [Saving a Copy of the Editor DRC Error View](#)

Viewing and Fixing Errors

The error browser displays errors in two views.

- To view detail routing errors or view verification errors from an error view, click the DRC tab.
- To view editor DRC errors in the current design, click the Editor DRC tab.

The error browser window contains, from top to bottom, an error hierarchy tree, an error list view, and an error detail view. You can use the split bars between to adjust the relative heights of these views.

- The error hierarchy tree displays the error view name and the number of total errors, visible errors, fixed errors, and null nets for each error view.
You can group the errors by type, the default, or by layer to help you find the problems you need to examine.
- The error list displays the ID number, error type, layer name, and a short summary for each error in the list.
- The error detail pane displays information about the errors that you select in the error list.
You can display information for up to 100 errors. If you hold the pointer over the error detail pane, the details for the first selected error appear in an InfoTip.

You can control the visual display of errors in the active layout view by choosing commands on the Options menu and setting options below the error detail pane.

To examine errors in the error browser,

1. (Optional) To group the errors by layer instead of type, choose Options > Show by Layer.
2. Select an error group in the error hierarchy tree.

To expand an error view, error type, or layer in the hierarchy, click its expansion button (+). When you select an error group, only the errors in that group appear in the error list.

3. (Optional) Filter the error data to display just the errors you need to examine.

The error data filter applies to all the errors in the error browser, including both detail routing error records and open error views.

To set the error data filter, choose Errors > Set Error Data Filter and set the filter criteria in the Set Error Data Filter dialog box. You can display or hide errors based on matching attribute values (error types, layers, net types, and route types) or net associations. For more details, see “[Filtering Errors in the Error Browser](#)” on page A-98.

4. Select one or more errors in the error list.

The error browser displays information about the selected errors in the error detail pane. You can Ctrl-click or drag the pointer to select multiple errors.

5. Examine the errors in the layout view.

By default, when you select an error in the error list, the layout view zooms in to magnify the error shape and colors it in the select color, which is white by default. The error shapes for the other errors in the error list appear in the DRC violation color, which is yellow by default.

You can highlight error shapes in the layout view by clicking buttons or choosing commands on the Highlight menu in the error browser window. You can highlight selected errors, the errors that are visible in the error list, specific types of errors or errors on specific layers, and errors associated with specific nets. For details about highlighting error shapes, see the “Highlighting Routing or Verification Errors” topic in IC Compiler Help.

You can adjust the layout view error display by setting options in the area below the error detail pane

- To control whether the tool displays all, selected, or no errors in the layout view, select an option in the Show list. The default is All.

- To control whether the tool zooms in, pans, or neither to display the selected errors, select an option in the Follow list. The default is Zoom.

To set the scaling factor for the Zoom option, select or type a value in the box beside the Follow list. The default scaling factor is 1.0.

- To control whether the tool dims other objects in the layout view when errors are displayed, select or deselect the Dim option. This option is deselected by default.

To further examine the errors, you can

- Select error shapes in the active layout view by using the Error Selection tool. For details, see “[Selecting Errors in the Layout View](#)” on page A-100.
- Display error information on the Query panel by clicking an error shape with the Query tool. For details about using the Query tool, see “[Viewing Object Information](#)” on page A-31.
- Highlight the nets for selected detail routing errors by choosing Options > Highlight Nets For Selected Errors. For details, see “[Examining Nets Associated With Detail Routing Errors](#)” on page A-101.
- Display the net nodes for selected Open Locator type errors by choosing Options > Display Net Nodes For Selected Open Locators. For details, see “[Examining Net Nodes Associated With Open Locator Errors](#)” on page A-101.

6. (Optional) Mark errors as fixed in the error list by selecting the errors and clicking the Fixed button.

You can display or hide fixed errors in the error list by choosing Options > Hide Fixed Errors.

You can save the updated error status indicators. Detail route error status changes are saved or discarded with other design changes when you save or close the design. The tool also automatically saves error views with unsaved status changes when you save the top-level design. You can manually save individual error views with unsaved status changes by using the Save Error Views dialog box. For details, see “[Saving Error Updates in an Error View](#)” on page A-103.

7. Examine other errors in the error list by repeating steps 4 through 6, or examine errors in a different error group by repeating steps 2 through 6.

The error browser is a global tool: errors that you select in the active layout view are cross-selected in any other open layout views. However, only the active layout view changes when the error browser automatically zooms or pans to display the selected errors. For more information, see the “Examining Errors in the Error Browser” topic in IC Compiler Help.

Filtering Errors in the Error Browser

You can filter the errors in the error browser to display just the errors that you want to examine. This error data filter can be based on matching attribute values, net associations, or both. You can either display or hide errors that match the filter criteria you specify in the Set Error Data Filter dialog box.

When you apply a filter, it sets a current filter state in the error browser that persists when you select different error views or groups of errors in the error hierarchy tree. The filter state also applies to errors that you load after setting the filter. Error filtering is not incremental; if you apply a new filter, the tool clears the previous filter.

When you filter the errors based on matching attribute values, you select the attribute values for the errors that you want to display or hide. You can filter the error views and detail routing records by

- Error type
- Layer
- Net type
- Route type

Note:

The error type and layer name attributes are available in the detail routing records and in all error views. The route type attribute is available only in categorized signoff DRC error

views, which are created by the `signoff_drc` command with the `-categorize_errors true` option. The net type attribute is available for error views that are associated with nets, including categorized signoff DRC error views and layout-versus-schematic (LVS) error views, and for detail routing errors.

You can filter the errors based on any attribute for which more than one value exists in the open error views, and you can select multiple values for each attribute. The Set Error Data Filter dialog box displays only the attribute values that are present in the detail routing errors or the open error views. For example, if the errors do not contain any shorts, the dialog box does not display the Short error type option.

When you filter the errors based on associated nets, you can select the nets or type the net names in the Set Error Data Filter dialog box. You can also type part of a net name or name pattern and allow the dialog box to complete the name to its longest match.

To set an error data filter,

1. Choose Errors > Set Error Data Filter.

The Set Error Data Filter dialog box appears. If you need assistance using this dialog box, click Help to display a Help page in the man page viewer.

2. Select a filter option.

The “Show only matching errors” option is selected by default. To hide the errors that match the filter criteria and display all the other errors, select the “Hide matching errors” option.

3. (Optional) To set filter criteria based on attribute values, select options as needed.

You can select any combination of error type, layer, net type, and route type options. To select all the options for an attribute, click All. To clear all the options for an attribute, click None.

To filter errors that are not associated with a net, select the No Net option. To filter errors that are not associated with a route type, select the No Route Type option.

If the open error views contain many error types or layers, the Set Error Data Filter dialog box displays scrolling lists instead of check boxes for these attributes. You can Ctrl-click items in the list to select or deselect error types or layers, and you can Shift-click items to select groups of error types or layers.

4. (Optional) To set filter criteria based on net associations, select the nets or type the net names in the “Specify nets” box.

You can type multiple net names by separating them with blank spaces. If you type an invalid net name, the name appears on a light red background.

You can type part of a net name and press Tab to complete the name to its longest match. If multiple names match the text, a name completion list appears. Double-click a name, or select one or more names and press Enter, to paste the names into the

“Specify nets” box. You can use the Up Arrow and Down Arrow keys to move up and down the list. To close the list, press Escape.

Note:

For optimal performance, the tool checks the names only after the first time you press Tab, press Enter, or display the name completion list for a specific object type. This means the invalid name indicator might not appear if you have not yet performed any of these operations.

5. Click Apply or OK.

The filter remains set until you modify it, remove it, or end the session.

To remove the filters, you can

- Click the Clear Data Filter button or choose Errors > Clear Data Filter in the error browser window.
- Click the Clear Filter button in the Set Error Data Filter dialog box.

For more information, see the “Filtering Errors in the Error Browser” topic in IC Compiler Help.

Selecting Errors in the Layout View

When you examine errors in the error browser, you can select routing or verification errors in the active layout view by using the Error Selection tool. This tool is available only when the All option, the default, is selected in the Show list. Error selection is local to the error browser and the active layout view and does not affect the global selection in other views.

To enable the Error Selection tool in the error browser,

- Click the  button in the error browser window.

The pointer becomes the Error Selection Tool pointer.

You can select or deselect error shapes in the active layout view.

- To select an error and deselect any other selected errors, click the error shape.
- To add an error to the current selection, Ctrl-click the error shape:
- To remove an error from the current selection, Shift-click the error shape:

You can select or deselect multiple errors at the same time by dragging the pointer to draw a rectangular box around the error shapes.

- To select the errors in a rectangular area and deselect any other selected errors, drag the pointer to draw a rectangular box around the error shapes.
- To add the errors in the area to the current selection, press the Ctrl key while you drag the pointer.
- To remove the errors in the area from the current selection, press the Shift key while you drag the pointer.

Examining Nets Associated With Detail Routing Errors

When you examine detail routing errors in the error browser, you can set an option to highlight the nets in the layout view that are associated with the errors selected in the error list.

To highlight the nets for selected detail routing errors,

1. Select the Detail Route error type in the error hierarchy tree.
2. Select one or more errors in the error list.
3. Choose Options > Highlight Nets of Selected Errors.

A check mark appears beside the command on the Options menu when this option is enabled.

Examining Net Nodes Associated With Open Locator Errors

When you examine open locator errors in the error browser, you can set an option to display the net nodes in the layout view that are associated with the errors selected in the error list.

To display the net nodes for selected open locator errors,

1. Select the Open Locator error type in the error hierarchy tree.
2. Select one or more errors in the error list.
3. Choose Options > Display Net Nodes for Selected Open Locators.

A check mark appears beside the command on the Options menu when this option is enabled.

Examining Errors in an Area

To analyze the relationships among errors due to their physical proximity, you can display just the errors that occur in a specific area of the design.

To display errors in an area of the design,

1. Select the error view or group of errors in the error hierarchy tree that contains the errors you want to examine.
2. Make sure the All option is selected in the Show list.
3. Display the area of the design you want to examine by using the Zoom In, Zoom Out, and Pan tools.
4. Enable the Error Selection tool, and then select errors in the area by clicking or dragging the pointer.
5. Filter the errors by clicking the Filter button, selecting the “Show only matching errors” option in the Set Error Data Filter dialog box, and clicking OK.

When you are ready to analyze errors in a different area, clear the error filter by clicking the Clear Filter button, and repeat steps 3 through 5.

Saving the Error List in an Error Report File

When you examine errors in the error browser, you can save the error list information as a report in a text file, and you can choose whether to exclude or include hidden errors.

To save the error list information in a report file,

1. Select the error group you want to save in the error hierarchy tree.
2. Choose one of the following commands on the Options menu:
 - To exclude hidden errors, choose Options > Save to File > Omit Hidden Records.
 - To include hidden errors, choose Options > Save to File > Include Hidden Records.The Save Errors to File dialog box appears.
3. Navigate to the directory where you want to save the file, and type the file name in the “File name” box.
4. Click Save.

Saving Error Updates in an Error View

The tool also automatically saves error views with unsaved status changes when you save the top-level design. You can manually save individual error views with unsaved status changes by using the Save Error Views dialog box. For example, if you opened the top-level design for read-only, you cannot save changes in the design, and you can save changes in the error views only by using the Save Error Views dialog box or the `save_mw_cel` command.

To save error views with error status updates,

1. Choose File > Save in the error browser window.

The Save Error Views dialog box appears. The dialog box lists the error views that have unsaved changes.

2. Select the error views you want to save.
3. Click the Save Selected button.

The tool saves the selected error views. Unsaved changes in error views that you do not select remain in memory.

The Save Error Views dialog box appears automatically if you close or reopen an error view that has unsaved changes. In this case, the dialog box provides a Discard button.

- Click the Save Selected button to save the changes in the selected error views.
- Click the Discard button to discard the unsaved changes in the selected error views.

Click Cancel if you want to return to the error browser without closing or reopening the error view. In this case, the tool does not save or discard the error view changes.

You can also save error view changes by entering the `save_mw_cel` command on the `icc_shell` command line. Note that if you close error views by entering the `close_mw_cel` command without first saving the error views, the tool discards any unsaved changes without notice.

The tool saves detail route error data changes automatically when you save the top-level design or discards the changes when you close the design without saving the design data. Because the detail route error data is in the top-level design instead of in an error view, you cannot save detail route error data changes by using the Save Error Views dialog box.

Saving a Copy of the Editor DRC Error View

The default editor DRC error view name is *design_cell_name_edrc.err*, where *design_cell_name* is the name of the current design. You cannot save the default editor DRC error view, and the tool does not save the error records when it closes the error view. However, you can save a copy of the editor DRC error view with a different name.

To save the editor DRC error view with a different name,

1. Choose File > Save As in the error browser window.
The Save Error View As dialog box appears.
2. Type an error view name, or click the browse button and select an error view in the file browser that appears.
The default name is *design_cell_name_edrc.err*, where *design_cell_name* is the name of the current design.
3. (Optional) To update a previous version of the error view in the same file, select the “Overwrite existing version” option.
4. Click OK.

You can load a saved copy of the editor DRC error view by choosing File > Open in the error browser window and selecting the Editor DRC option in the Open error view dialog box. However, editor DRC does not update loaded copies of the editor DRC error view incrementally when new violations are detected or old violations are fixed.

Editing the Physical Layout

The layout window provides interactive tools and commands for editing your physical design. You can edit the floorplan and object placement, create and remove physical objects, and route nets and physical buses in the active layout view.

You can use general-purpose editing tools and commands to

- Move, resize, copy, and remove objects
- Rotate and flip objects, and change cell orientations
- Reshape objects
- Align, distribute, and spread objects
- Move pins along an edge of a plan group or soft macro, pads along an edge of the die area, or terminals along an edge of the die area
- Expand objects

You can also use commands on the Floorplan menu to define or draw physical objects such as placement blockages, bounds, plan groups, pin guides, route guides, routing blockages, tracks, and voltage areas.

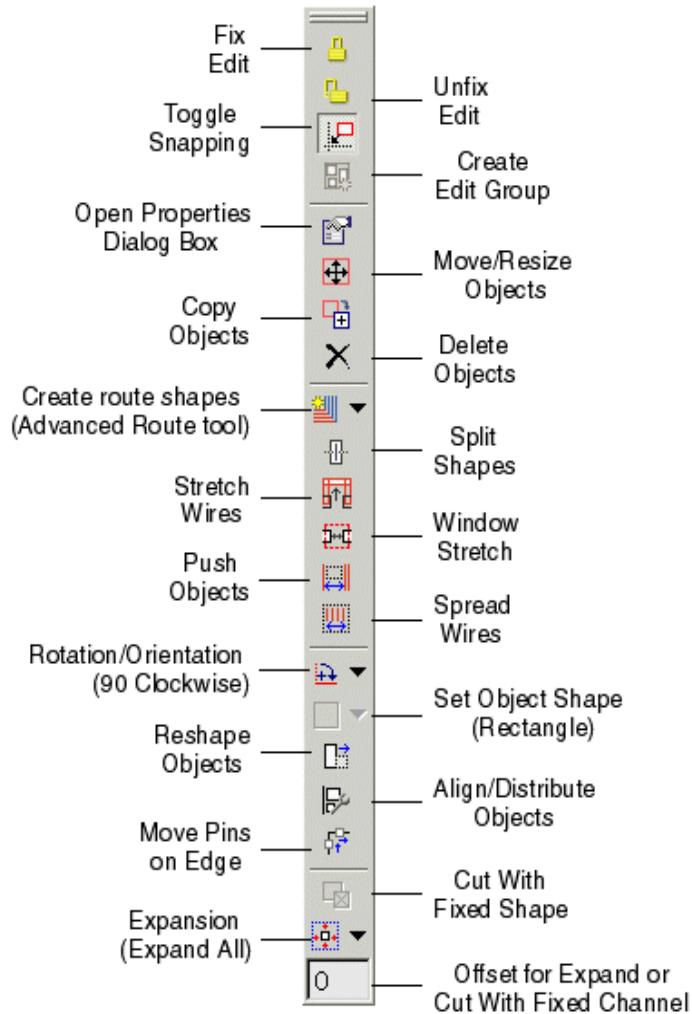
You can use route editing tools to

- Route physical buses and multiple nets
- Route individual nets
- Route individual nets on the redistribution layer in a flip-chip design
- Create routing corridors
- Create net and user shapes, vias and via arrays, port terminals, and text objects
- Create custom wires
- Split shapes
- Stretch wires while optionally maintaining their physical connections
- Move and stretch objects while optionally maintaining their physical connections
- Push objects away from an area while maintaining their physical connections
- Spread wires evenly on a layer while maintaining their physical connections

Typically, you select the objects that you want to edit. You can select the objects interactively with the Selection tool or by typing their names on the Select By Name toolbar. The interactive editing tools also allow you to select objects when the tool is active.

You can access editing tools and commands by clicking buttons on the Edit toolbar or by choosing commands on the Edit menu. [Figure A-24](#) shows the Edit toolbar as it appears when you select one or more objects in the layout view. The Edit toolbar groups related editing functions together in lists that open when you click the  button.

Figure A-24 Edit Toolbar When Objects Are Selected



For information about editing physical design objects and constraints, see the following sections:

- [Editing Objects Interactively](#)
- [Working in the Editing Environment](#)
- [Finding and Fixing Editor DRC Violations](#)
- [Editing Hierarchical Cells in Place](#)
- [Annotating the Layout](#)

Editing Objects Interactively

The layout editing mouse tools allow you to interactively create, edit, or remove physical objects in a layout view. The layout window provides a consistent editing environment for all of the tools. You can

- Set tool options on the Mouse Tool Options toolbar
- Use keyboard shortcuts for frequent or repetitive operations
- Preview objects for selection and cycle through overlapping objects

If you need to magnify or traverse the design or measure distances when an editing tool is active, you can temporarily activate the Zoom In tool, Zoom Out tool, Pan tool, or Ruler tool without deactivating the editing tool. For more details, see “[Selecting a Mouse Tool](#)” on [page A-26](#).

When you activate a mouse tool in a layout view, the options for using the tool appear on the Mouse Tool Options toolbar. See [Figure A-5 on page A-31](#) for an example of this toolbar.

[Table A-3](#) shows the buttons that you can use to perform actions with the active editing tool.

Table A-2 Layout Editing Tool Conventions

Action	Button
Finish operation and apply changes	
Cancel pending input, if any, or exit tool	
Reset toolbar options values to defaults	
Switch between toolbar and dialog box	
Show Help page for the active tool in the man page viewer	

You can perform frequent or repetitive actions by using keyboard shortcuts. [Table A-3](#) shows the standard key bindings for the interactive editing tools.

Table A-3 Interactive Editing Tool Conventions

Action	Shortcut
Add points to perform the operation	Click or Drag
Add point and apply changes	Return

Table A-3 Interactive Editing Tool Conventions (Continued)

Action	Shortcut
Remove last point	Backspace
Cancel pending input, if any, or exit tool	Escape
Cycle forward through overlapping objects	F1
Cycle backward through overlapping objects	Shift+F1
Reverse (undo) most recent operation	Ctrl+X
Reapply (redo) most recently applied operation	Ctrl+Y

In addition, if a tool needs to change layers, for example, to add a wire jog, you can press F1 or Tab to cycle through the layers.

Many of the editing tools operate on selected objects. You can select the objects before or after you activate a tool. Selected objects appear in the selection color, which is white by default.

You can preview an object before selecting it by holding the pointer over its position in the active layout view. Information about the object appears in an InfoTip. For details, see “[Previewing Objects](#)” on page A-28. If any objects are selected when you activate an editing tool, you can Ctrl-click objects to add them to the current selection and Shift-click objects to remove them from the current selection.

Note:

If you have trouble selecting or previewing small objects, try magnifying the view. In a large design, some objects or object outlines are too small to click at low magnification. For details, see “[Magnifying and Traversing the Design](#)” on page A-37.

All of the operations that you can perform with the interactive editing tools are logged and can be replayed with the `gui_mouse_tool` command. Similarly, options that you set on the Mouse Tool Options toolbar are logged and can be replayed with the `gui_set_mouse_tool_option` command. For information about these commands and the options for a particular tool, click the  button to see the Help page for that tool.

Most of the editing tools can provide interactive DRC with immediate visual feedback when the tool detects a DRC violation. When you use these tools with the editor DRC facility enabled, the layout view displays a different visual cue for each type of violation. These visual indicators allow you to avoid making changes that cause new DRC violations, but the tools do not prevent you from making these changes. For more details, see “[Finding and Fixing Editor DRC Violations](#)” on page A-126.

The Advanced Route tool, Create Route tool, Create RDL Route tool, Stretch Wire tool, and Window Stretch tool support nondefault routing rules by default. If you disable nondefault routing rule compliance, the tools use the rules defined in the technology file.

When you activate the Advanced Route tool, Create Route tool, or the Create RDL Route tool, the most frequently used tool options appear on the Mouse Tool Options panel. You can open a dialog box that provides all of the options for the tool. You can switch between the toolbar options and the dialog box options by clicking the  button.

For information about the interactive editing operations that you can perform, see the following sections:

- [Moving, Resizing, Modifying, and Removing Objects](#)
- [Expanding and Rotating Objects](#)
- [Routing Nets and Physical Buses](#)
- [Creating Floorplan Objects](#)

Moving, Resizing, Modifying, and Removing Objects

You can move, resize, copy, split, reshape, and remove objects. To perform basic move and resize operations without regard to connected wires, use the Move/Resize tool. If you need to maintain connectivity when you move or resize objects, use the Stretch Wire tool or the Window Stretch tool.

[Table A-4](#) shows the editing tools you can use to modify the physical design.

Table A-4 Editing Tools for Design Objects

To perform this operation	Use this tool
Move objects	Click  or choose either Edit > Move/Resize or View > Mouse Tools > Edit Tool, and then select the objects and click or drag to the new location. Alternatively, you can use the <code>move_objects</code> command.
Resize objects	Click  or choose either Edit > Move/Resize or View > Mouse Tools > Edit Tool, and then select an object and drag an edge or corner to resize. Alternatively, you can use the <code>resize_objects</code> command.
Copy objects	Click  or choose either Edit > Copy or View > Mouse Tools > Copy Tool, and then select and drag the objects. Alternatively, you can use the <code>copy_objects</code> command.

Table A-4 Editing Tools for Design Objects (Continued)

To perform this operation	Use this tool
Remove objects	Click  or choose Edit > Delete, and then click or drag a rectangle to delete objects. Alternatively, you can use the <code>remove_objects</code> command.
Split routing or user shapes	Click  or choose Edit > Split, and then drag a line or rectangle across the objects where the split is to occur. You can set options as needed on the Mouse Tool Options toolbar. Alternatively, you can use the <code>split_objects</code> command.
Stretch wires while optionally maintaining connectivity and adhering to nondefault routing rules	Click  or choose Edit > Stretch Wire, and then drag the wires. You can set options as needed on the Mouse Tool Options toolbar.
Move and stretch objects while optionally maintaining connectivity and adhering to nondefault routing rules	Click  or choose Edit > Window Stretch or View > Mouse Tools > Window Stretch Tool, and then drag the objects. You can set options as needed on the Mouse Tool Options toolbar.
Reshape objects by cutting or adding an area with an optional gap	Click  or choose Edit > Reshape, optionally set the gap size, and then drag a rectangle across the object where you want to cut or add a piece. You can set options as needed on the Mouse Tool Options toolbar.
Align objects relative to a side or center of an anchor object or location	Click  or choose Edit > Align, click the Align tab on the Mouse Tool Options toolbar, select the objects you want to align, specify the anchor object or location, and click the toolbar button for the alignment operation you want to perform. You can set options as needed on the Mouse Tool Options toolbar.
Distribute objects relative to a side of an anchor object or location	Click  or choose Edit > Distribute, click the Distribute tab on the Mouse Tool Options toolbar, select the objects you want to distribute, specify the anchor object or location, and click the toolbar button for the distribution operation you want to perform. You can set options as needed on the Mouse Tool Options toolbar.

Table A-4 Editing Tools for Design Objects (Continued)

To perform this operation	Use this tool
Spread objects relative to a side of an anchor object or location	Click  or choose Edit > Spread, click the Spread tab on the Mouse Tool Options toolbar, select the objects you want to spread, specify the anchor object or location, and click the toolbar button for the spread operation you want to perform. You can set options as needed on the Mouse Tool Options toolbar.
Move pins on an edge of a plan group or soft macro, pads on an edge of the die area, or terminals on an edge of the die area	Click  or choose Edit > Move Pins on Edge, and then select the objects and click or drag to the new location. Alternatively, you can use the <code>move_pins_on_edge</code> command.

For more information about using these editing tools, see the “Selecting an Editing Tool” topic in IC Compiler Help.

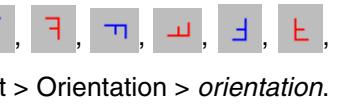
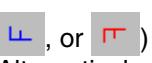
Expanding and Rotating Objects

[Table A-5](#) explains how you can expand, rotate, and flip design objects and orient cells.

Table A-5 Editing Commands for Design Objects

To perform this operation	Do this
Expand selected objects to objects on one or all sides	Click an expansion tool ( ,  ,  ,  , or ) or choose Edit > Expand > <i>expansion_direction</i> . Alternatively, you can use the <code>expand_objects</code> command.
Rotate selected objects 90 or 180 degrees	Click a rotation tool ( ,  , or ) or choose Edit > Rotate > <i>rotation</i> . Alternatively, you can use the <code>rotate_objects</code> command.

Table A-5 Editing Commands for Design Objects (Continued)

To perform this operation	Do this
Flip selected objects on an x- or y-axis	Click a flip tool ( or ) or choose Edit > Rotate > axis. Alternatively, you can use the <code>flip_objects</code> command.
Orient selected objects to north	Click an orientation tool ( , or ) or choose Edit > Orientation > orientation. Alternatively, you can use the <code>rotate_objects</code> command.

For more information about these editing commands, see the “Expanding Objects,” “Rotating Objects,” and “Changing Cell Orientations” topics in IC Compiler Help.

Routing Nets and Physical Buses

Table A-6 shows how you can use interactive editing tools to create route objects and route nets and buses in the physical design.

Table A-6 Interactive Route Editing Tools and Utilities

To perform this operation	Use this tool
Route physical buses or multiple nets by optionally adhering to nondefault routing rules and interactive DRC rules	Click  or choose Edit > Create > Advanced Route Tool, select the buses or nets, and click or drag to create route segments. You can set options as needed on the Mouse Tool Options toolbar or in the Advanced Route Tool dialog box.
Route single nets by optionally adhering to nondefault routing rules and interactive DRC rules	Click  or choose Edit > Create > Route Tool, select the net, and click or drag to create route segments. You can set options as needed on the Mouse Tool Options toolbar or in the Create Route Tool dialog box.
Route single nets on a redistribution layer by optionally adhering to nondefault routing rules and interactive DRC rules	Click  or choose Edit > Create > RDL Route Tool, select the net, and click or drag to create route segments. You can set options as needed on the Mouse Tool Options toolbar or in the Create RDL Route Tool dialog box.

Table A-6 Interactive Route Editing Tools and Utilities (Continued)

To perform this operation	Use this tool
Create new routing corridors and add rectangles to existing routing corridors	Click  or choose Edit > Create > Routing Corridor, set the minimum and maximum layers and the rectangle width, optionally specify one or more nets, and click or drag to draw the rectangles. You can set options as needed on the Mouse Tool Options toolbar or the Create Routing Corridor Tool dialog box.
Push objects away from a rectangular area on a layer while maintaining connectivity and optionally adhering to nondefault routing rules	Click  or choose Edit > Area Push, select a shape type, select a layer, and click or drag the pointer to draw the rectangular area. You can set options as needed on the Mouse Tool Options toolbar.
Spread wires evenly by layer between two points while maintaining connectivity and optionally adhering to nondefault routing rules	Click  or choose Edit > Spread Wires, select a shape type, select two or more wires, and click or drag the pointer to specify the points. You can set options as needed on the Mouse Tool Options toolbar.
Add missing vias to selected wires or to wires on selected nets	Select the wires or nets and choose Edit > Route Utilities > Add Missing Vias. Alternatively, you can use the <code>gui_add_missing_vias</code> command.
Trim dangling wire ends on selected wires or on all the wires of selected nets	Select the wires or nets and choose Edit > Route Utilities > Trim Dangling Wires. Alternatively, you can use the <code>gui_trim_dangling_wires</code> command.
Create net shapes and user shapes	Click  or choose Edit > Create > Shape, select a shape type, select a net (net shape only), and click or drag to draw the shape. You can set options as needed on the Mouse Tool Options toolbar.
Create vias and via arrays	Click  or choose Edit > Create > Via, select a via master or enable automatic mode, drag to the wire intersection, and click to insert the via or via array. You can set options as needed on the Mouse Tool Options toolbar.

Table A-6 Interactive Route Editing Tools and Utilities (Continued)

To perform this operation	Use this tool
Create terminals	Click  or choose Edit > Create > Terminal, select a port, and click or drag to draw the terminal. You can set options as needed on the Mouse Tool Options toolbar.
Create text objects	Click  or choose Edit > Create > Text, type the text string and set options as needed on the Mouse Tool Options toolbar, and click or drag to place the text in the layout view.
Route one or more nets by adhering to preroute DRC rules	Click  or choose Edit > Create > Custom Wires, select the nets, and click or drag to create wire segments. You can set options as needed in the Create Custom Wires dialog box.

For more information about these tools, see “[Routing Nets and Buses in the GUI](#)” on [page 6-135](#) and the “Selecting an Editing Tool” topic in IC Compiler Help.

Creating Floorplan Objects

You can use the GUI or `icc_shell` commands to create floorplan objects in the physical design.

Table A-7 shows the floorplan objects you can create and the menu commands and the `icc_shell` commands you can use to create them.

Table A-7 Create Floorplan Object Commands

Object	Menu command	icc_shell command
Placement blockages	Floorplan > Create Placement Blockage	<code>create_placement_blockage</code>
Bounds	Floorplan > Create Bound	<code>create_bounds</code>
Plan groups	Floorplan > Create Plan Group	<code>create_plan_groups</code>
Route guides	Floorplan > Route Guide	<code>create_route_guide</code>
Routing blockages	Floorplan > Routing Blockage	<code>create_routing_blockage</code>
Routing tracks	Floorplan > Create Track	<code>create_track</code>

Table A-7 Create Floorplan Object Commands (Continued)

Object	Menu command	icc_shell command
Voltage areas	Floorplan > Create Voltage Area	create_voltage_area
Pin guides	Pin Assignment > Create Pin Guide	create_pin_guide

For more information about creating these objects, see the “Creating Floorplan Objects” topic in IC Compiler Help.

Working in the Editing Environment

The editing environment provides an incremental undo and redo facility that allows you to reverse or reapply the most recent editing operation or a series of operations. You can also constrain (fix) objects during editing operations, control grid snapping, set editing options, define and edit object groups, view and edit object properties, and enable or disable editor DRC for interactive design rule checking during editing operations. [Table A-8](#) explains the global operations that you can use with any editing tool or command.

Table A-8 Global Editing Operations

To perform this operation	Use this tool
Reverse the previous edit change	Click the  button or choose Edit > Undo. Alternatively, you can use the <code>undo</code> command.
Reapply the subsequent edit change	Click the  button or choose Edit > Redo. Alternatively, you can use the <code>redo</code> command.
Constrain (fix) selected objects to prevent changes from interactive edits	Click the  button or choose Edit > Constraints > Fix Edit. Alternatively, you can use the <code>set_object_fixed_edit</code> command.
Remove (unfix) the editing constraint on selected objects to allow changes from interactive edits	Click the  button or choose Edit > Constraints > Unfix Edit. Alternatively, you can use the <code>set_object_fixed_edit</code> command.
Enable or disable grid snapping	Click the  button or choose Edit > Snapping. Alternatively, you can use the <code>set_object_snap_type</code> command with the <code>-enabled</code> option.

Table A-8 Global Editing Operations (Continued)

To perform this operation	Use this tool
Create an edit group that contains the objects in the current selection	Click the  button or choose Edit > Create Edit Group. Alternatively, you can use the <code>create_edit_group</code> command.
View the estimated resistance values for wires, vias, and pins on a net	Choose Route > Report Estimated Resistance.
View or edit object attribute values for selected objects.	Click the  button or choose Edit > Properties.

Note:

For details about using the Properties dialog box, see “[Viewing and Editing Object Properties](#)” on page [A-42](#).

For information about the physical design editing environment, see the following sections:

- [Reversing and Reapplying Edit Changes](#)
- [Fixing and Unfixing Objects for Edits](#)
- [Changing the Way Objects Snap to a Grid](#)
- [Setting Editing Options](#)
- [Editing Groups of Objects](#)
- [Viewing Estimated Resistance Values for Wires, Vias, and Pins](#)

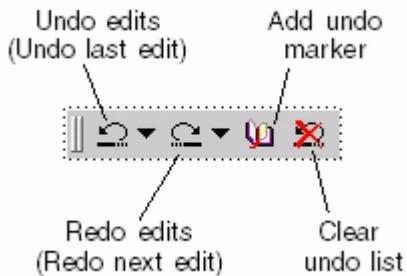
Reversing and Reapplying Edit Changes

You can reverse (undo) or reapply (redo) many of the interactive editing operations that you perform in the IC Compiler GUI. You can reverse the most recent editing operation or a series of operations. If you reverse one or more editing operations, you can reapply the most recently reversed operation or a series of operations. You can also add markers to the undo list that let you quickly reverse all subsequent editing operations.

Note:

When you use a command that is not reversible, the tool automatically clears the undo list. You can no longer reverse previous editing operations, and the Undo command is dimmed on the Edit menu.

The Undo toolbar shown in [Figure A-25](#) lets you reverse or reapply the most recent operation or a series of operations, add markers to the Undo list, and clear the Undo list.

Figure A-25 Undo Toolbar

- To reverse the most recent editing operation, click the button on the Undo toolbar, press **Ctrl+Z**, choose **Edit > Undo**, or enter `undo` on the command line.
- To reapply the most recently reversed editing operation, click the button on the Undo toolbar, press **Ctrl+Y**, choose **Edit > Redo**, or enter `redo` on the command line.

You can reverse any editing operation and all subsequent operations by selecting the operation you want to reverse. Similarly, you can reapply any editing operation that you reversed and all subsequently reversed operations by specifying the operation you want to reapply.

- To reverse a specific editing operation and all subsequent operations, choose the operation on the Undo menu (on the Undo toolbar). To reverse all the operations in the undo list, choose **<Undo All>**.
- To reapply a specific editing operation and all subsequently reversed operations, choose the operation on the Redo menu (on the Undo toolbar). To reapply all of the operations in the redo list, choose **<Redo All>**.

You can add markers to the undo list to indicate the beginning or end of a series of operations that you might want to reverse or reapply together. When you add a marker, the tool displays “<mark>” in the undo list. If you choose a marker to reverse the subsequent operations, the tool displays the marker in the redo list.

- To add a marker to the undo list, click the button on the Undo toolbar.
- To clear the undo and redo lists, click the button on the Undo toolbar.

Fixing and Unfixing Objects for Edits

You can fix objects in the active layout view to

- Prevent editing change
- Use as anchors for the distribution or splitting of multiple objects.

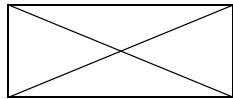
To fix objects,

1. Select the objects to fix.

The objects become highlighted.

2. Click the  button on the Edit toolbar or choose Edit > Constraints > Fix Edit.

Each fixed object displays an X across it.



Fixed



Not fixed

Note:

When you use the Move/Resize tool to move or resize objects, you can allow the tool to edit fixed objects by selecting the “Ignore fixed” option on the Mouse Tool Options toolbar.

To unfix objects and allow editing changes,

1. Select the objects to unfix.
2. Click the  button on the Edit toolbar or choose Edit > Constraints > Unfix Edit.

Alternatively, you can allow or prevent editing for an object by changing the `is_fixed` attribute value in the Properties dialog box.

Changing the Way Objects Snap to a Grid

When you create or edit objects interactively in the layout view by using an editing tool or command, IC Compiler first modifies the objects and then snaps them to the nearest associated snap grid. When object snapping is enabled, IC Compiler automatically moves objects to the grid that is associated with the object type. You can enable or disable snapping for all editing operations or for individual object types, and you can select a different snap grid for each type of object. When snapping is disabled, the tool snaps the objects to the lithography (Min) grid.

To enable or disable snapping for all object operations,

- Click the  button on the Edit toolbar or choose Edit > Snapping.

The toolbar button and the icon beside the Snapping command on the Edit menu indicate whether snapping is enabled or disabled.

Alternatively, you can click the  button on the Mouse Tool Options toolbar when an interactive editing tool is active. You can also enable or disable snapping by using the `set_object_snap_type` command with the `-enabled` option. For example, to enable object snapping when it is disabled, enter

```
icc_shell> set_object_snap_type -enabled true  
0
```

When you use the `-enabled` option, the command returns the previous snapping state: 0 for disabled or 1 for enabled.

To change the ways objects are snapped,

1. Choose Edit > Options > Snap Settings.

The Snap Settings dialog box appears. If you need assistance using this dialog box, click Help to display a Help page in the man page viewer.

2. Enable snapping for all object types if necessary by clicking the “Enable snapping” option.
3. Enable or disable snapping for individual object types as needed by clicking the options to the left of the object type names.
4. Change the snap grid for individual object types by selecting an option in the list to the right of the object type name.

The tool provides the following snap grid options. Note that the list for a particular object type displays only the applicable options for that object type.

- Row and Min Grid
- Row and Unit Tile
- Row and Placement Site
- Middle Row and Min Grid
- Middle Row and Unit Tile
- Middle Row and Placement Site
- Routing Track

- Middle Routing Track
- User Grid

Alternatively, you can change the snap grid by using the `set_object_snap_type` command with the `-class` and `-snap` options. For example, to snap hard macro objects to the user grid, enter

```
icc_shell> set_object_snap_type -class hard_macro -snap user  
litho
```

When you use the `-snap` option, the command returns the previous snap type. For more information about the `set_object_snap_type` command, including lists of the valid object classes and snap types, see the man page.

5. Set other options as needed.

- To snap pin shapes to the nearest edges of their parent objects, select the “Snap pin to parent edge” option.
- To snap objects to the nearest horizontal or vertical edge of the nearest visible, selectable object, select the Snap Object Edge option.

When edge snapping is enabled, you can set the search radius, in pixels, by typing a value in the “Edge search radius” box. The search radius controls how far the tool searches for the nearest edge. A small search radius allows finer control. A larger search radius allows snapping over longer distances.

- To snap objects to routing tracks only in the preferred direction, select the “Snap to preferred tracks only” option.

This option is deselected by default, which means that objects can snap to tracks in both the preferred and nonpreferred directions.

When you are ready to close the Snap Settings dialog box, click Close.

Setting Editing Options

When you use an interactive editing tool in the active layout view, you can control how the tool operates by setting options on the Mouse Tool Options toolbar. You can also set some of the more frequently used editing options by using the Edit Options dialog box. In addition, you can use the Edit Options dialog box to set expand object hit type options, connectivity options, and specific editing tool options.

When you change an editing tool option, either on the Mouse Tool Options toolbar or in the Edit Options dialog box, the change occurs immediately for all applicable tools in all open layout views.

To open the Edit Options dialog box,

- Choose Edit > Options > Edit Options.

The Edit Options dialog box appears. If you need assistance using this dialog box, click Help to display a Help page in the man page viewer.

To set frequently used editing tool options,

1. Click the General tab in the Edit Options dialog box.

This tab is selected by default.

2. Set options as needed.

- To retain the placement of child objects such as the cells associated with a plan group or soft macro, select the Keep Placement option.

- To retain a constant distance between I/O pads and the core area when you resize the core area or the die area, select the “Keep Core to Pad” option.

This option is selected by default. Deselect this option if you want to let the tool change the distance between the core area and the I/O pads.

- To edit fixed objects, select the “Ignore fixed attribute” option.

An object is fixed when the `is_fixed` attribute value is `true`.

- To let the editing tools ignore wire end caps and use the bounding box to define the wire geometry, select the “Ignore end caps” option.

If you split or cut a wire when this option is selected, the tool adjusts the center line and makes an exact cut of the bounding box. In addition, if you change the end cap from an extended type to a nonextended type, the tool adjusts the center line to maintain the same bounding box.

This option is deselected by default, which means the editing tools do not ignore end caps and use the center line to define the wire geometry. If you split or cut a wire, the tool cuts the center line, which can change the bounding box of the line. In addition, the bounding box might change if you change the end cap from an extended type to a nonextended type.

- To let the Delete tool remove disconnected vias when it removes the associated wire or path shape, select the “Remove disconnected vias” option.

- To make sure the net shapes have the same route type when an editing tool merges two net shapes, select the “Check route types compatible when merging” option.

- To prevent the layout view from automatically displaying the object types and layers for objects that you modify by using an editing tool, deselect the “Auto display edited objects and layers” option.

This option is selected by default, which means that if you modify an object that is not visible because its object type or layer is hidden, the layout view automatically displays the hidden objects or layer.

- To prevent the layout view from automatically displaying object types and layers for routing objects that you create by using an editing tool, deselect the “Auto display previewed objects and layers” option.

This option is selected by default, which means that if you create an object that is not visible because its object type or layer is hidden, the layout view automatically displays the hidden objects or layer.

To set expand object hit type options,

1. Click the Expand tab in the Edit Options dialog box.
2. Select or deselect expand object hit type options as needed,

These options control which object types are used as targets when you expand objects. You can use soft macros, hard macros, plan groups and bounds, placement blockages, polygons, and voltage areas. If all of these options are deselected, objects expand to the edge of the core area. For more details, see the “Expanding Objects” topic in IC Compiler Help.

To set connectivity options,

1. Click the Connectivity tab in the Edit Options dialog box.
2. Set options as needed.

- To prohibit the Stretch Wire tool or Window Stretch tool from editing wires that have staggered via connections, deselect the “Allow staggered via connections” option.

This option is selected by default, which means the tools can edit these wires.

Deselect this option if you want to avoid poor editing results that can be caused by staggered via connections.

A staggered via is a stack of two or more connected vias with center points that are not coincident. The tools can cause opens or nonoptimal connections by trying to repair the vias at these multiple connection points.

- To allow the Stretch Wire tool and Window Stretch tool to edit connections across multiple layers, select the “Constrain edit propagation to single layer” option.

This option is selected by default, which means connectivity edits are constrained to connections across a single layer or on the same layer. If wire jogging is enabled, the tool creates jogs instead of editing wires and vias on other layers.

- To make sure that existing vias match the vias that you add to selected wires or wires on selected nets, select the “Update existing vias for add missing vias” option.

When this option is selected, the tool tries to match existing vias on the selected wires or nets to the types of vias it would create if no vias existed at those connection points. Vias that match are retained. Vias that do not match are removed and replaced with new vias to complete the connections.

Note that the tool removes any extra vias at the connection points because they have no match.

- To display the connectivity of selected objects automatically when you edit them with the Stretch Wire tool or Window Stretch tool, select the “Automatically show connectivity” option.

To set options for specific editing tools,

1. Click the Mouse Tools tab in the Edit Options dialog box.
2. Set options as needed.

- To allow the Stretch Wire tool to change wire widths, select the “Resize wires” option.
- To constrain wires to a specific width on a layer when you modify wire widths with the Stretch Wire tool, select the “Snap width resize to discrete values” option.

This option is deselected by default and is available only when the “Resize wires” option is selected.

To define the list of permitted wire widths on a layer, use the `gui_set_layer_widths` command. To view the list of permitted wire widths for a layer, use the `gui_get_layer_widths` command.

- To force the Stretch Wire tool to skip occupied tracks and move the selected wire to the nearest free track when you press an arrow key to move the wire to the next track, select the “Snap wire to next free track” option.

A free track is a track that has no overlapping wires on the same layer when the wire is placed on the track.

If you want to set the maximum search distance that the Stretch Wire tool can use to find a free track, type a value in the Search Distance box. If the tool does not find a free track within the search distance, it does not move the wire. Note that if the “Snap Width Resize to Discrete Values” option is selected, the tool searches for a track that matches the width of the wire you are moving.

Editing Groups of Objects

You can interactively edit multiple objects at the same time by defining the objects as an edit group. To define an edit group, you assign a name and specify the objects. An edit group consists of one or more geometric objects and maintains the relative positions and orientations of the objects during editing operations.

In the layout view, a geometric object has a bounding box and can be moved to a new position. The bounding box of an edit group is the union of all the object bounding boxes and is dynamically calculated and updated during editing operations. An object can belong to only one edit group, and an edit group cannot contain another edit group.

To define a group of objects for editing,

1. Select the objects.

2. Click the  button on the Edit toolbar or choose Edit > Create Edit Group.

The tool assigns a unique name to the edit group, selects the group, and displays the bounding box in the layout view.

In general, an edit group forms a loose coupling of the objects. You can still manipulate them as separate objects. The edit group just helps you to perform the editing task.

For more information about using edit groups, see the “Editing Groups of Objects” topic in IC Compiler Help.

Viewing Estimated Resistance Values for Wires, Vias, and Pins

You can use the Report Estimated Resistance dialog box to examine a report of the estimated resistance values for wire segments (net shapes), vias, and pins on a routed net. By default, the report displays the estimated resistance values for each wire segment and via and the estimated pin-to-pin resistance values for each net.

You can select multiple objects in the report to view their combined estimated resistance value. You can also filter the report to display wire segments only or pins only. As you select the layout objects in the table, the tool magnifies the design to fit the selected objects in the layout view and displays them in the current highlight color. You can set options to preserve or clear the highlighting and to disable the zoom-to-fit mechanism. You can also change the highlight color or enable the automatic color cycling mechanism.

By default, the tool calculates sheet resistance and via resistance as defined in the technology file. To compute resistance by using the TLU+ file, click the Options button and select the Use TLU+ option.

To generate the estimated resistance report for a net,

1. Choose Route > Report Estimated Resistance.

The Report Estimated Resistance dialog box appears. If you need assistance using this dialog box, click Help to display a Help page in the man page viewer.

2. Select the net or type the net name in the Net box.

You can select the net or an object associated with a net (such as a net shape or pin) before or after you open the Report Estimated Resistance dialog box.

3. (Optional) If you need to adjust the per unit resistance values for resistance estimation, click the Options button and set values as needed in the Set Resistance per Layer dialog box.

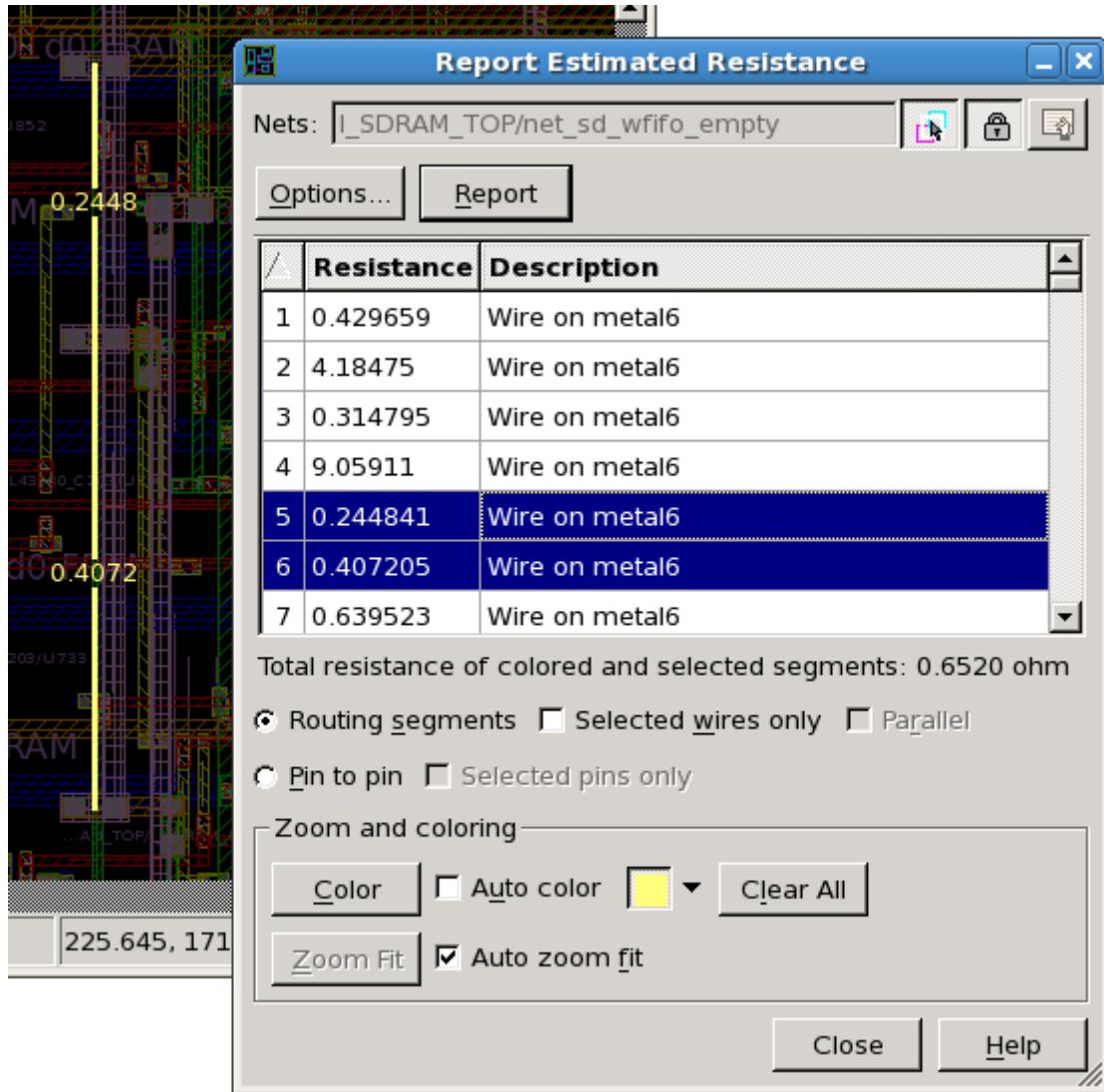
If the technology file does not include these values, the default value is 0. In this case, you must provide the per unit resistance values.

4. Click the Report button.

The tool displays the total estimated resistance value for the objects below the report table in the dialog box and the estimated resistance value for each object in the layout view.

[Figure A-26](#) shows an example of the Report Estimated Resistance dialog box with two wire segments selected in the report table. The total estimated resistance value for the selected objects appears below the report table.

Figure A-26 Example of an Estimated Resistance Report for a Net



For more information about viewing estimated resistance values for a net, see the “Viewing Estimated Wire Resistance” topic in IC Compiler Help.

Finding and Fixing Editor DRC Violations

When you edit or create objects interactively in the layout view, you can check the nets in the design for routing design rule violations. The layout view provides an editor DRC facility that allows you to find and fix DRC violations caused by interactive editing operations. Editor DRC detects design rule violations, displays error shapes in the active layout view, and records the violations in an editor DRC error view.

You can enable editor DRC for interactive editing operations or run editor DRC on nets that you edited while editor DRC was disabled. Editor DRC provides rule checking for a limited set of advanced DRC rules defined in the technology file.

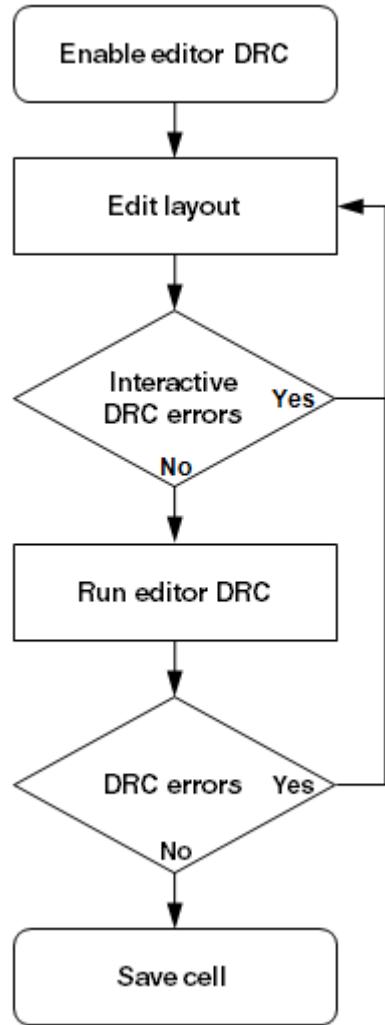
Important:

For optimal performance, editor DRC runs only a quick check and displays errors using a limited set of the DRC rules defined in the technology file. It does not fix the errors or guarantee that the editing tools create legal placement or routes without DRC violations. After editing your design interactively, run the `check_legality` command and a DRC or LVS verification tool to identify errors that you need to fix. For more details, see [“Performing Design Rule Checking Using the IC Compiler Tool” on page 6-120](#).

The Advanced Route tool, Copy tool, Create Route tool, Create Shape tool, Create Via tool, Move/Resize tool, Reshape tool, Split tool, Stretch Wire tool, and Window Stretch tool can use editor DRC to detect violations and provide immediate visual feedback while you perform editing operations. The Area Push tool and Spread Wire tool do not allow you to make changes that cause DRC violations. However, these tools check for violations after you apply the changes because the changes might fix existing violations.

You control which design rules editor DRC checks by setting options in the Editor DRC Options dialog box. [Figure A-27](#) shows the flow of a route editing session using editor DRC.

Figure A-27 Route Editing Flow Using Editor DRC



For more information about using editor DRC, see the following sections:

- [Verifying Route Edits Interactively](#)
- [Verifying Edited Nets](#)
- [Setting Editor DRC Options](#)
- [Examining Editor DRC Errors](#)

Verifying Route Edits Interactively

To enable or disable the editor DRC for interactive editing operations,

- Choose Edit > Editor DRC > Enable DRC.

Alternatively, you can select or deselect the DRC option on the Mouse Tool Options toolbar.

When you enable editor DRC for an interactive editing operation, it provides immediate visual feedback if it detects a violation. The layout view displays a different visual cue for each type of violation. These visual indicators allow you to avoid making changes that cause new DRC violations, but the tools do not prevent you from making these changes.

Editor DRC supports rule checking on the following types of objects:

- Horizontal and vertical net shapes and user shapes, including wires, vias, via arrays, paths, polygons, and rectangles
- Port shapes and terminals
- Standard cells, hard and soft macros, and black boxes
- Routing blockages

In a flip-chip design, you can control whether right angle connections and acute angle connections are permitted for 45-degree routes on the redistribution layer. By default, editor DRC permits right angle connections but marks acute angle connections as violations.

When nondefault routing rule compliance is enabled, editor DRC records nondefault routing rule violations as nondefault routing rule related errors. If you disable nondefault routing rule compliance, editor DRC removes the errors related to nondefault routing rules from the error view.

For more information, see the “Checking Edits for DRC Violations” topic in IC Compiler Help.

Verifying Edited Nets

If you perform interactive editing operations while editor DRC is disabled in the active layout view, you should check the edited net shapes for editor DRC violations.

To check selected nets for editor DRC violations,

- Select one or more nets you need to check.
- Choose Edit > Editor DRC > Run DRC on Selected Nets.

Alternatively, you can use the `gui_check_drc_errors` command.

To check one or more individual nets for editor DRC violations,

1. Choose Edit > Editor DRC > Run DRC.

The Run DRC on Nets dialog box appears. If you need to reset the dialog box options to their defaults, click Default. For assistance running editor DRC, click Help to display a Help page in the man page viewer.

2. Type the net names in the nets box. Separate the net names with blank spaces. Alternatively, you can select the nets.

3. Select an option to check entire nets or net shapes in a region.

To check all the shapes on the specified nets, select the “Check entire nets” option.

To check just the shapes within a rectangular area, select the “DRC area” option and define the area. You can either type the x- and y- coordinates for the upper left and lower right corners or draw area in the layout view.

4. To restrict the rule checking to nets on specific layers, select the “Restrict layers” option and select or deselect layer names as needed.

5. Set other options as needed.

- To check for opens on the specified nets, select the “Check for opens” option.

This option is available only when the “Check entire nets” option is selected.

- To check for nondefault routing rule violations, select the “Honor NDR” option.

- To restrict the rule checking to shorts between the specified nets, select the “Report only violations between specified nets” option.

- To change the maximum number of errors that the tool can report, type a value in the “Maximum number of errors to report” box.

The default number of errors is 250.

- To change the maximum number of net shapes that the tool can check, type a value in the “Maximum number of net shapes to check” box.

The default number of net shapes is 10,000.

6. Click OK or Apply.

Alternatively, you can use the `gui_check_drc_errors` command.

For more information, see the “Checking Edits for DRC Violations” topic in IC Compiler Help.

Setting Editor DRC Options

Editor DRC provides rule checking for a limited set of advanced technology DRC rules defined in the technology file. The interactive editing tools use editor DRC to detect routing design rule violations that occur during interactive editing operations.

Editor DRC automatically checks for violations of the following design rules:

- Metal spacing
- Shorts
- Fat metal width spacing
- Via corner spacing
- Special end-of-line spacing
- Special notch

You can set options to enable rule checking for the following design rules:

- Via density
- Minimum edge
- Minimum length and area
- Enclosed via spacing
- General via spacing

In a flip-chip design, you can control whether right angle connections and acute angle connections are permitted for 45-degree routes on the redistribution layer. By default, editor DRC permits right angle connections but marks acute angle connections as violations.

You can also set the maximum number of violations that the tool can detect, and you can control whether the tool opens the error browser and loads the errors automatically when violations are detected.

For descriptions of the rules that editor DRC can check, click the Help button in the Editor DRC Options dialog box.

To set editor DRC options,

1. Choose Edit > Options > DRC Options.

The Editor DRC Options dialog box appears. If you need to reset the dialog box options to their defaults, click Default. For assistance using this dialog box, click Help.

2. Set advanced rule checking options as needed.
3. Set RDL routing rule checking options as needed for a flip-chip design.
4. Set other options as needed.

When you change an option setting in the Editor DRC Options dialog box, the change occurs immediately for all applicable tools in all open layout views; If GUI command output logging is enabled in the Application Preferences dialog box, the tool also records the change by displaying a `gui_set_pref_value` command in the console log view and history view.

For more information, see the “Setting Editor DRC Options” topic in IC Compiler Help.

Examining Editor DRC Errors

When editor DRC detects design rule violations, it displays error shapes in the active layout view, records the errors in an editor DRC error view, and automatically loads the error view into the error browser editor DRC view. For information about using the error browser, see “[Examining Routing and Verification Errors](#)” on page A-94.

The error view contains error records for violations detected during interactive editing operations and violations detected when you run checks on specific nets or on nets within a rectangular area. Editor DRC incrementally updates the error view when you fix a violation, reverse an editing operation that caused a violation, disable the design rule check used to detect a violation, or remove errors in the error browser.

The editor DRC error view is associated with the current design. It remains open and loaded in the error browser until you close the design or change the current design. However, the tool does not save the error view when you save or close the design. If you close the current design, editor DRC closes the error view and removes the errors from the error browser.

For more information about using editor DRC and examining editor DRC errors, see the “[Examining Editor DRC Errors](#)” topic in IC Compiler Help.

Editing Hierarchical Cells in Place

You can traverse the design hierarchy to view or edit the contents of blocks and hierarchical cells within the context of the primary design in a layout window. The edit-in-place facility allows you to view and edit subdesign CEL views and FRAM views without closing the primary design.

Note:

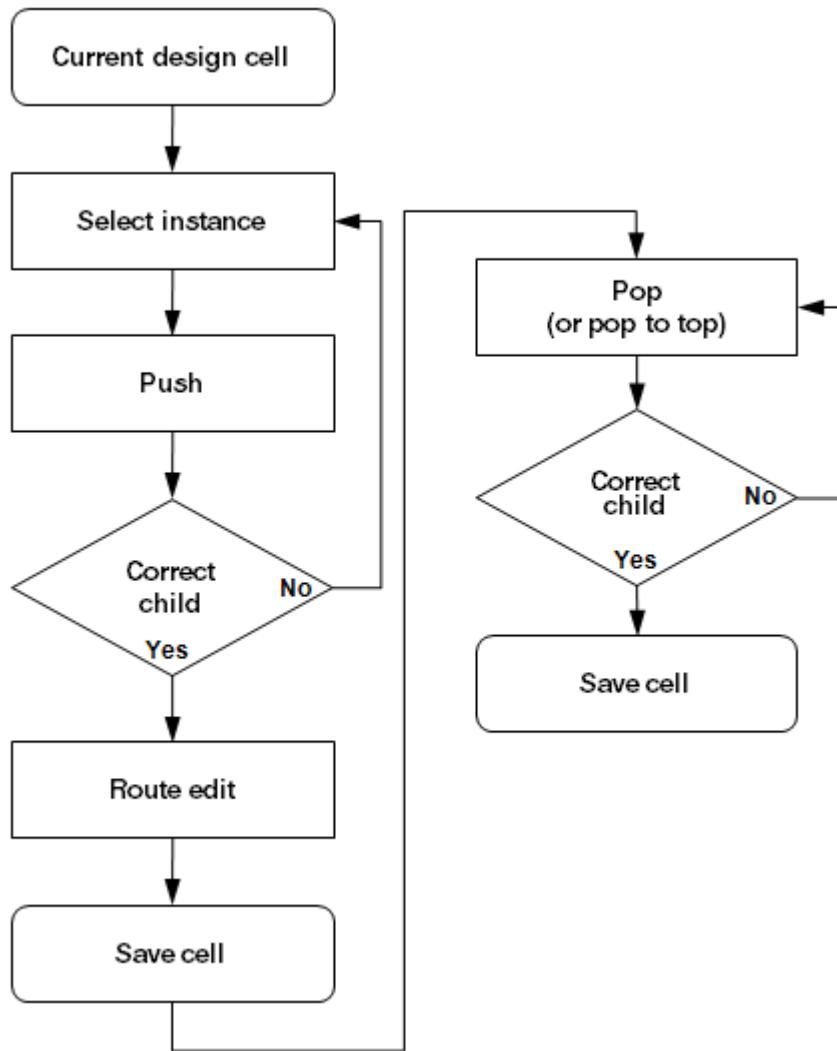
For information about the primary design in a layout window, see “[Setting the Primary Design](#)” on page [A-22](#).

The edit-in-place push and pop operations are similar to using the `open_mw_cel` and `current_mw_cel` commands to navigate the hierarchy. However, the edit-in-place operations also maintain the relationships between the cells and help synchronize parent cells to edited child cells.

- *Push* opens a child cell on the next hierarchy level below the current design, if it is not already open, and changes the current design to that cell.
- *Pop* changes the current design to the parent cell on the next hierarchy level above the current design.

[Figure A-28](#) shows the edit-in-place workflow.

Figure A-28 Edit-In-Place Workflow

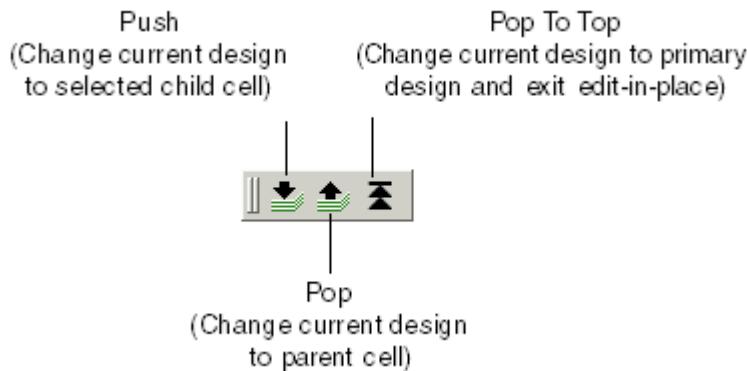


The cell in which the layout window editing environment is currently active is known as the *current edit cell*. You can perform any available GUI or `icc_shell` operation in the current edit cell. During an edit-in-place session, all of the layout view tools and commands, such as highlighting, flylines, editing tools, and visual modes operate within the context of the current edit cell.

You can select and edit objects in the current hierarchy level and subhierarchy levels specified by the level box in the View Settings panel. When the current edit cell changes, the tool clears the undo stack. Flylines and net connectivity lines for selected objects appear only in the current edit cell. For example, if you select a port on the current edit cell, the layout view displays flylines to show the connections from the port to pins in the cell, but it does not display flylines to show connections from the port to ports or pins in the parent cell.

You can perform edit-in-place operations by using tools on the Edit In Place toolbar shown in [Figure A-29](#).

Figure A-29 Edit In Place Toolbar



To enable the edit-in-place facility and change the current design to a cell one level down in the design hierarchy,

1. Select the cell you need to edit.
2. Click the button on the Edit In Place toolbar or choose Edit > In Place > Push.

Alternatively, you can use the `change_working_design` command with the `-push` option.

Note:

To edit a soft macro cell, the soft macro must reside in the design library and the design library must be open for reading and writing. If you open a cell in a library that is open for reading only, the tool automatically opens the cell for reading only.

To change the current design to the parent cell of the current edit cell,

- Click the button on the Edit In Place toolbar or choose Edit > In Place > Pop.
- The tool automatically synchronizes the cell boundaries and port instances with the corresponding boundaries and pins in the edited cell.

Alternatively, you can use the `change_working_design` command with the `-pop` option.

Important:

The tool does not save or close the child cell. To save your edits at the end of an edit-in-place session, choose File > Save Design or use the `save_mw_cell` command to save the top-level design with all the subdesigns in the hierarchy.

To change the current design to the top-level cell and disable the edit-in-place facility,

- Click the  button on the Edit In Place toolbar or choose Edit > In Place > Pop to Top.

Alternatively, you can use the `change_working_design -pop` command.

For more information about using the `change_working_design` command to traverse the hierarchy and open cells for editing, see “[Opening Designs in a Hierarchical Stack](#)” on [page 3-17](#).

By default, the primary design remains visible in the layout view during an edit-in-place session, but the tool dims the objects on hierarchy levels above the current edit cell and displays the objects inside the cell at full brightness.

- The zoom and pan setting does not change.

To magnify the design and fit the current edit cell in the layout view, choose View > Zoom > Zoom Fit All or press F. To shrink the design and fit the primary design in the layout view, choose View > Zoom > Zoom Fit Top or press Ctrl+F.

- The global selection list does not change when you change the current edit cell.

However, the layout view displays the selection color only on objects in the current edit cell.

The layout view displays highlight, visual mode, and map mode colors only in the current edit cell.

- The legend on the Visual Mode panel displays the data for the design in which you loaded it. You must reload the data to display data for the current edit cell.
- The legend on the Map Mode panel displays the data for the current edit cell automatically. However, if no map data exists for a cell, the layout view clears the legend on the Map Mode panel.

The layout view displays rulers only in the current edit cell. Rulers in other open cells are hidden.

GUI tools display coordinates relative to the origin of the current edit cell, such as ruler coordinates or the pointer position displayed on the status bar. When you change the current edit cell, the GUI records a `change_working_design` command with the cell instance name in the session log. This allows you to determine the local origin for any coordinates specified by commands in the current edit cell.

You can set options on the View Settings panel to control the type of view the tool opens, the context in which the current edit cell appears, and the brightness of objects outside the current edit cell.

To set edit-in-place options for the active layout view.

1. On the View Settings panel, click the Settings tab, and then click the View tab.
 2. Set options as needed.
 - To change the type of view that the tool opens, select an option in the Child View Name list. The valid choices are DEFAULT, FRAM, and CEL. The default is DEFAULT.
 - To display the current edit cell by itself in the layout view, deselect the “Edit in Place” option.
- By default, this option is selected and the layout view displays the current edit cell in its place within the primary design. If you deselect this option, the layout view displays the current edit cell in its normal orientation regardless of its orientation within the primary design. For information about the primary design in a layout window, see [“Setting the Primary Design” on page A-22](#).
- To set the brightness level for objects outside the current edit cell when the “Edit in Place” option is selected, select an option in the brightness () list.
3. Click Apply.

If you open multiple layout views in the same layout window, all the layout views have the same current edit cell. If you want to edit different cells at the same time, you must open them in different layout windows.

If the error browser is open when you change the current edit cell, the tool automatically loads the editor DRC error view for the new current design, if it exists, into the error browser editor DRC view. Only error views associated with the current design can be loaded in the error browser. However, regardless of which design is the current design, the layout view continues to display of error shapes for any design that is open in the layout window.

If Hercules VUE is open when you change the current edit cell, the layout view continues to display DRC errors in the primary design. However, you should load the errors for the current edit cell if you want the layout view to interpret coordinates from Hercules VUE in the correct coordinate space.

For more information about using the edit-in-place facility in a layout window, see the “Editing Hierarchical Cells in Place” topic in IC Compiler Help.

Annotating the Layout

You can create or remove annotation objects in the active layout view. You can also create or remove collections of annotations and change annotation attributes.

To create an annotation, use the `gui_add_annotation` command and specify the location. You can use options to specify the annotation attributes such as the annotation text, object

shape, color, fill pattern, line width, and line style. For example, you can enter the following command to create a rectangular green annotation shape:

```
icc_shell> gui_add_annotation -type rect \
    -color green {{200 200} {777 777}}
```

The command returns a collection that contains the new annotation.

You can create an annotation group by specifying a group name. You can also specify the instance name of the layout window. For example, you can enter the following command to add an annotation to the group named Test1 in the layout window named Layout.2:

```
icc_shell> gui_add_annotation -window Layout.2 -group Test1 \
    -type polyline -color red -width 2 {{123 456} {789 12} {200 200}}
```

If you specify preview text and query text when you create an annotation, you can preview and query the annotation object in the layout view.

- To specify preview text for an annotation, use the `-info_tip` option.
- To specify query text, use the `-query_text` option. You must specify preview text if you specify query text.

For example, you can enter the following command to add an annotation with preview and query text:

```
icc_shell> gui_add_annotation -info_tip {My preview text} \
    -query_text {My query text} {{10.5 2.0} {23.5 14.6}}
```

In the layout view, the tool displays the annotation in the preview color when you hold the pointer over the annotation. The preview text is displayed in an InfoTip. When you click the annotation with the Query tool, the tool displays the query text on the Query Object panel. If you do not specify query text for the annotation, the preview text appears on the Query Object panel.

You can also define the InfoTip and query text by specifying a Tcl procedure. This technique generates text dynamically for the InfoTip or query text based on the current state of the design, rather than using a static string. To use a Tcl procedure to generate the text, begin the query text or InfoTip text with an equal sign (=). The remainder of the string is treated as the Tcl command or procedure to invoke when the InfoTip or query text is displayed. The `gui_add_annotation` command recognizes the following special strings and replaces the text with the appropriate value when the command is called.

Table A-9 gui_add_annotation Attributes

Attribute	Description
<code>%window</code>	The layout window name
<code>%view</code>	The layout view name

Table A-9 gui_add_annotation Attributes

Attribute	Description
%object	A collection containing the annotation
%x	x-coordinate of the mouse
%y	y-coordinate of the mouse

The following example defines a Tcl procedure named `annotation_query` that returns the `points` attribute for the current annotation. The `-query_text` option of the `gui_add_annotation` command references the `annotation_query` Tcl procedure. Note that the procedure name is preceded by an equal sign. When you click the annotation, the tool displays the points for the annotation in the Query Object panel.

```
proc annotation_query { object window view } { \
    return [get_attribute $object points] \
}

gui_add_annotation -window Layout.1 -type rect -color blue \
    -query_text {=annotation_query %object %window %view} \
    -info_tip {Annotation points} {{100 100} {200 200}}
```

You can also invoke a user-defined Tcl procedure when you click the annotation by specifying the `-query_command` option followed by the Tcl procedure and options. The following example defines a Tcl procedure named `annotation_cmd` and references the procedure by using the `gui_add_annotation` command. The procedure prints a message to the console. Note that the `-query_command` option does not require a leading equal sign (=) for the Tcl procedure name.

```
proc annotation_cmd { object window view x y } {
    puts "Clicked annotation $object $window $view $x $y"
}

gui_add_annotation -window Layout.1 -type rect -color red \
    -query_command {annotation_cmd %object %window %view %x %y} \
    -info_tip "Run annotation_cmd" {{200 200} {300 300}}
```

To create a collection of annotations, use the `gui_get_annotations` command. For example, to create a collection of the annotations within a rectangular area, you can enter the following command:

```
icc_shell> gui_get_annotations -within {{10.5 2.0} {23.5 14.6}}
```

The command returns a collection of the annotations that match the values of the options that you specify. You can use the `set_attribute` command to change annotation attributes.

You can remove annotations in the layout view by using the `gui_remove_annotations` command.

- To remove an annotation group, use the `gui_remove_annotations` command with the `-group` option.
- To remove a collection of annotations, use the `gui_remove_annotations` command with the `anno` argument, which specifies the collection that you want to remove.

You can remove all the annotations by using the `gui_remove_all_annotations` command.

Index

A

add missing vias A-113
add piece to object A-110
add_clock_drivers command 5-111
add_end_cap command 7-53
add_tap_cell_array command 7-3
adjust_premesh_connection command 5-109
advanced editing tools
 Advanced Route tool A-112
 Area Push tool A-113
 Spread Wire tool A-113
Advanced Library Format (ALF) 8-14
Advanced Route tool 6-140, A-112
ALF format 8-14
Align/Distribute tool A-110, A-111
aligning objects A-110
alignment tool A-110
all_rp_groups command 9-89
all_rp_hierarchicals command 9-89
all_rp_inclusions command 9-89
all_rp_instantiations command 9-89
all_rp_references command 9-89
all_scenarios command 3-38
always-on
 power wells, multivoltage designs 12-21
synthesis

exclusive move bounds 12-21
multivoltage designs 12-40
Analysis toolbar A-70
analysis tools, layout
 flylines A-70, A-71
 map modes A-70, A-75
 net connectivity A-70, A-72
 visual modes A-70, A-72
analyze_subcircuit command 5-118
antenna rules
 checking 7-22
 disabling 7-22
 enabling 7-21
antenna violations
 fixing
 disabling 7-22
 enabling 7-21
Application Preferences dialog box A-22, A-23
application task, layout window A-21
Area Push tool A-113
area utilization 4-56
area, voltage 3-24
attributes
 connected_to_iso_cell 12-20
 direct_power_rail_tie 12-35
 dont_touch 12-23, 12-24, 12-27
 ignore, removing 9-91
 is_isolation_cell 12-27, 12-28

is_level_shifter 12-27, 12-28
isolation_cell_enable_pin 12-28
level_shifter_enable_pin 12-28
relative placement, writing to disk 9-90
size_only 12-24, 12-25, 12-28

B

balance_inter_clock_delay command 5-103
blockages
 creating placement blockages 4-5
 creating routing blockages A-114
blockages, creating placement blockages
 A-114
boundary cell, clock tree synthesis 5-61
bounds, creating objects A-114
buffer trees
 creating 4-24
 removing 4-24
 reporting 4-24
buffering high-fanout nets 4-24
buffer-type level shifters
 inserting 12-23
 placing 12-27
bus routing tool A-112

C

Calibre interface 6-133
capacitance, net capacitance histogram A-88
capture window or view image A-55
categorized timing report interactive viewer
 4-67
CCS timing libraries
 scaling 3-46
cell orientations A-112
 displaying or hiding A-60
cell placement statistics 6-114
Cells List toolbar 3-17
change_names command 3-63

change_working_design command A-135,
 A-136
check_clock_tree command 5-82
check_isolation_cells command 12-28, 12-44
check_legality command A-127
check_level_shifters command 12-40
check_mv_design command 4-20, 12-27,
 12-43
check_noise command 8-8
check_physical_design command 3-25
check_rp_groups command 9-65
check_timing command 3-48
check_tlu_plus_files command 3-42
check_zrt_routability command 6-8
chip finishing
 adding end caps 7-53
 adding tap cell arrays 7-3
 displaying critical area heat maps 7-38
 filling empty tracks with metal fill 7-57
 inserting pad fillers 7-56
 inserting standard cell fillers 7-49
 inserting tap cells 7-5
 inserting well fillers 7-55
 performing wire spreading 7-39
 performing wire widening 7-40
Clear Rulers A-26, A-58
clock
 convergent 5-7
 overlapping 5-7
clock arrival histograms 5-148
clock graph view A-25
clock mesh
 adding mesh drivers 5-111
 analysis 5-118
 building premesh trees 5-111
 compiling premesh trees 5-114
 definition 5-105
 flow 5-107, 5-121
 prerequisites 5-106
 removing 5-110
 routing mesh nets 5-112

routing premesh trees 5-115
specifying 5-109
splitting clock nets 5-109
structure 5-105
clock network element tables, path inspector
 window A-92
clock sink group
 clock tree exceptions 5-14
 preexisting gates 5-15
 sink group definition 5-12
 sink group options 5-13
clock tree
 removing 5-80
 reporting 5-135, 5-136
 verifying 5-82
 viewing 5-143
clock tree arrival time histogram, displaying
 5-148
clock tree clustering, on-chip-variation-aware
 5-62
clock tree design rule constraint
 maximum capacitance 5-50
 maximum fanout 5-50
 maximum transition time 5-50
clock tree exceptions
 defining 5-17
 don't buffer net 5-28
 don't size cell 5-28
 don't touch subtree 5-27, 5-80
 exclude pin 5-11, 5-22
 float pin 5-23
 nonstop pin 5-22
 per-clock basis 5-17
 preserve hierarchy 5-29
 priority 5-21
 reporting 5-20
 size-only cells 5-29
 stop pin 5-11, 5-27
clock tree fanout browser 5-148
clock tree optimization
 ECO routing 5-103
 optimization capabilities
postroute, default 5-102
preroute, default 5-102
 selecting 5-102
postroute 5-102
preroute 5-102
timing analysis, router for 5-103
voltage area support 12-36
clock tree reference
 defined 5-4
 specifying 5-43
clock tree sinks
 default 5-11
 defining 5-11, 5-27
 optimizing 5-87
clock tree synthesis
 boundary cell, defined 5-61
 don't touch subtree
 defined 5-27, 5-80
 logic-level balancing 5-64
 exclude pin
 defined 5-11, 5-22
 implicit 5-11
 float pin
 defined 5-23
 examples 5-25
 ILM, logic-level balancing 5-64
 logic-level balancing
 defined 5-63
 don't touch subtree 5-64
 enabling 5-64
 hard macro 5-65
 ILM 5-64
 maximum capacitance 5-50
 maximum fanout 5-50
 maximum transition time 5-50
 minimum insertion delay, defined 5-53
 nonstop pin
 defined 5-22
 implicit 5-11
 prerequisites
 design 5-3
 library 5-4

skew, defined 5-53
stop pin
 defined 5-11, 5-27
 implicit 5-11
timing goals 5-53
voltage area support 12-36
clock tree synthesis options
 reporting 5-49
 resetting 5-49
clock_opt command 5-84, 12-34
clock_opt_feasibility
 command 5-143
close_mw_cel command A-103
closing the GUI A-2, A-23
command
 console, main window A-8
 execution at startup 2-2
 history A-15
 listing for session A-15
 listing in GUI A-12
 menu, choosing current task A-21
 saving session transcript A-15
 set_preroute_focal_opt_strategy 4-89
command line 2-3
 entering commands 2-4
 icc_shell 2-4
 shell 2-4
 X-term window 2-4
command log file
 record and replay GUI session A-23
 using 2-3
command.log file 2-3
commands
 add_clock_drivers 5-111
 add_end_cap 7-53
 add_tap_cell_array 7-3
 adjust_premesh_connection 5-109
 all_rp_groups 9-89
 all_rp_hierarchicals 9-89
 all_rp_inclusions 9-89
 all_rp_instantiations 9-89
 all_rp_references 9-89
 all_scenarios 3-38
 analyze_subcircuit 5-118
 balance_inter_clock_delay 5-103
 change_names 3-63
 change_working_design A-135, A-136
 check_clock_tree 5-82
 check_isolation_cells 12-28, 12-44
 check_legality A-127
 check_level_shifters 12-40
 check_mv_design 4-20, 12-27, 12-43
 check_noise 8-8
 check_physical_design 3-25
 check_rp_groups 9-65
 check_timing 3-48
 check_tlu_plus_files 3-42
 check_zrt_routability 6-8
 choosing in menus A-9
 clock_opt 5-84, 12-34
 clock_opt_feasibility 5-143
 close_mw_cel A-103
 commit_fp_plan_groups 9-57
 commit_skew_group 5-161
 compare_delay_calculation 3-49
 compile 12-5
 compile_clock_tree 5-99
 compile_premesh_tree 5-111, 5-114
 compile_ultra 12-5, 12-23
 copy_floorplan 3-21
 copy_objects A-109
 copying session log A-13
 create_bounds 4-13, A-114
 create_bounds, exclusive 12-21
 create_buffer_tree 4-24
 create_clock_mesh 5-109
 create_edit_group A-116
 create_mw_lib 3-5
 create_physical_bus 6-137, 6-140
 create_physical_buses_from_patterns 6-138
 create_pin_guide A-115
 create_placement 12-27
 create_placement_blockage A-114

create_plan_groups A-114
create_plan_groups command A-114
create_route_guide 6-28, A-114
create_routing_blockage 6-36, A-114
create_track A-114
create_voltage_area 12-4, 12-14, 12-16, A-115
create_zrt_shield 7-43
current_mw_cel 3-17, A-133
define_routing_rule 5-54, 6-16
define_scaling_lib_group 3-47
define_user_attribute A-50
derive_pg_connection 3-26
derive_placement_blockages 4-10
expand_objects A-111
extract_rc 6-146
extract_rp_group 9-101
flatten_clock_gating 5-108
flip_objects A-112
Follow Selection A-40
get_always_on_logic 12-41
get_bounds 4-16
get_cts_scenario 5-91
get_drc_errors 6-134
get_license 2-10
get_magnet_cells 4-83
get_placement_blockages 4-5, 4-10
get_route_guides 6-33
get_route_opt_zrt_crosstalk_options 6-67
get_routing_blockages 6-37
get_selection A-13
get_timing_paths A-86
Go Back in Zoom and Pan History A-40
Go Forward in Zoom and Pan History A-40
gui_add_missing_vias A-113
gui_check_drc_errors A-129, A-130
gui_get_layer_widths A-123
gui_mouse_tool A-108
gui_set_current_task A-21
gui_set_flat_hierarchy_color A-49
gui_set_layer_widths A-123
gui_set_mouse_tool_option A-108
gui_show_form A-10
gui_start A-2
gui_stop A-23
gui_trim_dangling_wires A-113
gui_unset_flat_hierarchy_color A-49
gui_write_layout_image A-56
help 2-5
history A-15
history view A-16
icc_shell 2-4
insert_pad_filler 7-56
insert_tap_cells_by_rules 7-5
insert_well_filler 7-55
keyboard shortcuts in GUI A-12
legalize_placement 12-27
license_users 2-10
list_drc_error_types 6-134
magnet_placement 4-80
map_retention_cell 12-29
move_objects A-109
move_pins_on_edge A-111
Named Zoom and Pan Settings A-40
open_mw_cel A-133
optimize_clock_tree 5-101
optimize_pre_cts_power 5-36
order_rp_groups 9-102
Pan to Selection A-37
place_opt 4-29, 12-4, 12-5, 12-27, 12-34
place_opt_feasibility 4-47
preroute_focal_opt 4-86
process_particle_probability_file 7-37
propagate_constraints 12-39
psynopt 12-4
read_def 3-20
read_drc_error_file 6-133
read_floorplan 3-21
read_parasitics 3-54
read_sdc 3-48
read_sdf 3-54
read_stream 3-65
read_verilog 3-14, 3-15
redo A-115, A-116, A-117

refine_placement 4-90
remove_all_spacing_rules 4-18
remove_annotations 3-54
remove_bounds 4-16
remove_buffer_tree 4-24
remove_clock_tree 5-80
remove_clock_tree_exceptions
 -dont_buffer_nets 5-28
 -dont_size_cells 5-28
 -dont_touch_subtrees 5-27
 -exclude_pins 5-22
 -float_pins 5-25
 -non_stop_pins 5-22
 -preserve_hierarchy 5-30
 -size_only_cells 5-29
 -stop_pins 5-27
remove_filler_withViolation 7-53
remove_from_rp_group 9-99
remove_ideal_network 3-48
remove_keepout_margin 4-5
remove_level_shifters 12-27
remove_license 2-11
remove_net_routing_layer_constraints 3-52
remove_objects A-110
remove_placement_blockage 4-10
remove_preferred_routing_direction 6-27
remove_route_guide 6-33, 6-37
remove_routing_blockage 6-37
remove_rp_group_options 9-91
remove_rp_groups 9-92
remove_sdc 3-48
remove_self_gating_logic 5-32
remove_skew_group 5-161
remove_stdcell_filler
 -pad 7-56
 -stdcell 7-53
 -tap 7-6
remove_target_library_subset 4-20
remove_voltage_area 12-21
remove_well_filler 7-56
report_bounds 4-16
report_buffer_tree 4-24
report_cell 12-44
report_clock_timing 5-139
report_clock_tree 5-135
report_clock_tree_power 5-32
report_constraint 5-167
report_critical_area 7-36
report_design 6-114
report_drc_error_type 6-134
report_error_coordinates 6-10
report_inter_clock_delay_options 5-43
report_keepout_margin 4-5
report_mw_lib 3-7
report_net_routing_layer_constraints 3-52
report_net_routing_rules 6-25, 6-26
report_noise_calculation 8-11
report_operating_conditions 12-45
report_physical_signoff_options 6-124
report_placement_utilization 4-56
report_power_guide 12-21
report_preferred_routing_direction 6-26
report_preroute_focal_opt_strategy 4-89
report_route_opt_strategy 6-94
report_route_opt_zrt_crosstalk_options 6-67
report_routing_rules 5-54
report_scan_chain 4-79
report_signal_em 8-17
report_signal_em_calculation 8-20
report_spacing_rules 4-18
report_target_library_subset 4-20, 12-45
report_timing 4-58, 5-167
report_timing_derate 3-46
report_tlu_plus_files 3-42
report_units 3-9, 8-15
report_voltage_area 12-21, 12-44
reset_design 3-48
reset_inter_clock_delay_options 5-42
reset_timing_derate 3-46
resize_objects A-109
reusing A-16
rotate_objects A-111, A-112
route_htree 5-115
route_mesh_net 5-112

route_opt 12-34
route_zrt_eco 6-110
route_zrt_group 6-72, 6-74
rp_group_inclusions 9-89
rp_group_instantiations 9-89
rp_group_references 9-90
save_mw_cel 3-60, A-103, A-135
saving transcript of session commands A-16
set_ahfs_options 4-24
set_checkpoint_strategy 4-45
set_clock_tree_exceptions
 -dont_buffer_nets 5-28
 -dont_size_cells 5-28
 -dont_touch_subtrees 5-27
 -exclude_pins 5-22
 -float_pin options 5-23
 -non_stop_pins 5-22
 -preserve_hierarchy 5-29
 -size_only_cells 5-29
 -stop_pins 5-27
set_clock_tree_options 6-24
set_clock_tree_references 5-43
set_congestion_options 4-12
set_delay_calculation_options 3-49
set_delay_estimation_options 3-51, 3-52,
 3-53, 3-54
set_em_options 8-18
set_extraction_options 6-147
set_gui_stroke_binding A-69
set_gui_stroke_preferences A-69
set_ignored_layers 6-13
set_inter_clock_delay_options 5-40
set_isolation 12-28
set_keepout_margin 4-4
set_left_right_filler_rule 7-52
set_level_shifter_strategy 12-23
set_level_shifter_threshold 12-23, 12-25,
 12-26
set_lib_cell_spacing_label 4-17, 4-18
set_max_transition 5-52
set_min_library 3-44
set_mw_lib_reference 3-10

set_net_routing_layer_constraints 3-52
set_net_routing_rule 6-25
set_object_fixed_edit A-115
set_object_snap_type A-115, A-118, A-119,
 A-120
set_operating_conditions 3-55, 12-24
set_optimize_dft_options 4-34
set_physical_signoff_options 6-124, 7-58
set_power_guide 12-21
set_power_net_to_voltage_area 12-39
set_preferred_routing_direction 6-26
set_route_opt_strategy 6-93
set_route_opt_zrt_crosstalk_options 6-66
set_rp_group_options 9-93
set_si_options 3-49, A-83
set_skew_group 5-160
set_spacing_label_rule 4-17, 4-18
set_target_library_subset 4-18
set_timing_derate 3-46
set_tlu_plus_files 3-41, 3-42, 7-57
set_write_stream_options 3-64
split_clock_gates 5-37
split_clock_net 5-109
split_objects A-110
spread_zrt_wires 7-39
stroke A-68
stroke activated commands A-68
uncommit_fp_soft_macros 9-57
undo A-115, A-116, A-117
uniquify_fp_mw_cel 3-15
unset_power_guide 12-21
update_clock_latency 5-104
verify_route 6-120
widen_zrt_wires 7-40
write_def 3-63
write_mw_lib_files 3-10
write_parasitics 3-64, 6-147
write_rp_groups 9-90
write_sdc 3-64
write_stream 3-64
Zoom Fit All A-37
Zoom Follows Selection A-37

Zoom In 2x A-37
Zoom Out 2x A-37
Zoom to Selection A-37
commit_fp_plan_groups command 9-57
commit_skew_group command 5-161
compare_delay_calculation command 3-49
compile command 12-5
compile_clock_tree command 5-99
compile_premesh_tree command 5-111, 5-114
compile_ultra command 12-5, 12-23
compression, relative placement 9-44
congestion map
 generating 6-114
 layer-based 6-114
congestion options 4-12
connected_to_iso_cell attribute 12-20
Connectivity Settings panel A-72
console
 command line A-12
 entering commands A-12
 history view A-15
 log view A-13
 main window A-8
 saving log messages A-14
 searching the transcript A-14
 window A-12
constraints
 reading 3-48
 report 5-167
 saving 3-64
 timing 3-47
constraints, voltage area 6-69
conventions for documentation xxxv
convergent clock path, defined 5-7
Copy tool A-109
copy_floorplan command 3-21
copy_objects command A-109
copying
 objects A-109
session log A-13
coupling capacitance
 extracting 6-146
 scaling 6-147
create_floorplan_object tools A-114
Create Net tool A-113
Create RDL Route tool A-112
Create Route tool 6-135, A-112
Create Routing Corridor tool A-113
Create Terminal tool A-114
Create Text tool A-114
Create Via tool A-113
create_bounds command 4-13, A-114
 exclusive 12-21
create_buffer_tree command 4-24
create_clock_mesh command 5-109
create_edit_group command A-116
create_mw_lib command 3-5
create_physical_bus command 6-137, 6-140
create_physical_buses_from_patterns
 command 6-138
create_pin_guide command A-115
create_placement command 12-27
create_placement_blockage command A-114
create_plan_groups command A-114
create_route_guide command 6-28, A-114
create_routing_blockage command 6-36, A-114
create_track command A-114
create_voltage_area command 12-4, 12-14, 12-16, A-115
create_zrt_shield command 7-43
creating floorplan objects A-114
critical area
 displaying heat maps 7-38
 encrypting and decrypting the particle probability function 7-37
crosstalk
 analysis
 delta delay visual mode 8-10

noise reports 8-11
preparation 8-7
static noise visual mode 8-11
timing report 8-10
defined 8-2
net lists, path inspector window A-92
optimization
 about 8-6
 postroute 8-6
prevention
 clock shielding 8-5
 clock tree synthesis 8-5
 placement 8-4
 track assignment 8-5
reduction, during route_opt 6-66
timing impacts 8-10
cts_do_characterization variable 5-135
cts_enable_rc_constraints variable 5-74
cts_fix_clock_tree_sinks variable 5-88
cts_move_clock_gate variable 5-87
cts_rc_relax_factor variable 5-74
cts_self_gating_connect_scan_enable
 variable 5-32
cts_self_gating_data_toggle_rate_filter
 variable 5-32
cts_self_gating_num_regs variable 5-32
cts_self_gating_num_regs_max_percent
 variable 5-32
cts_self_gating_num_regs_max_percent_total
 _cell variable 5-32
cts_self_gating_reg_power_filter variable 5-33
cts_self_gating_wns_backoff variable 5-33
cts_traverse_dont_touch_subtrees variable
 5-27
cts_use_debug_mode variable 5-135
cts_use_lib_max_fanout variable 5-51
cts_use_sdc_max_fanout variable 5-51
current design, setting 3-17
current edit cell A-133
current task, layout window A-21
current_mw_cel command 3-17, A-133
currentPrecision, in the technology file 11-7
Custom Wire tool A-114
customer support xxxvi
cut piece from object A-110

D

dangling wires, trimming A-113
database, Milkyway
 about 3-3
 saving a design 3-60
datapath, physical, defined 9-1
DEF file
 reading 3-20
 writing 3-63
define_routing_rule command 5-54, 6-16
define_scaling_lib_group command 3-47
define_user_attribute command A-50
delay calculation algorithm
 Arnoldi 3-49
 AWE 3-49
 Elmore 3-49
Delete tool A-110
densely-packed objects A-28
derating factors for timing library 3-46
derive_pg_connection command 3-26
derive_placement_blockages command 4-10
design for manufacturing (DFM)
 critical area heat map displaying 7-38
 inserting metal fill 7-57
 wire spreading 7-39
 wire widening 7-40
design overlays, layout view A-66
design rule violations
 analyzing
 DRC query commands 6-134
 error browser 6-134
designs
 schematics A-90
detail route options

descriptions 6-58
reporting 6-57
setting 6-57
detail routing
 running 6-86
dialog boxes
 finding A-10
direct_power_rail_tie attribute 12-35
DISPLAY environment variable
 checking before startup A-2
 overriding 2-2
distributing objects A-110
distribution tool A-110
domain, power 3-24
dont_touch attribute 12-23, 12-24, 12-27
draw rulers A-58
DRC, interactive editing errors A-126

E

ECO routing
 during clock tree optimization 5-103
ECO routing, running 6-110
edit groups A-124
Edit In Place toolbar A-135
Edit Options dialog box A-120
Edit Snapping Settings panel A-118
Edit tool A-109
Edit tool (see Move/Resize tool)
editing object properties A-42
editing tool options A-120
edit-in-place
 about A-133
 current edit cell A-133
 pop operation A-133
 push operation A-133
 setting options A-137
 workflow A-134
editor DRC A-115
 enabling A-129

gui_check_drc_errors command A-129,
 A-130
route editing flow A-126
setting options A-131
verifying edited nets A-129
edits
 redo A-116
 undo A-116
electromigration
 loading net switching information for analysis
 8-15
 performing signal analysis 8-17
enable level shifter 3-24
enable-type level shifters
 inserting 12-28
 placing 12-27
end cap
 adding 7-53
 defined 7-53
endpoint slack histogram
 histogram views A-87
 Timing Views toolbar A-87
environment variable, DISPLAY A-2
environment variables
 SNPS_MAX_QUEUETIME 2-11
 SNPS_MAX_WAITTIME 2-11
 SNPSLMD_QUEUE 2-10
error browser
 about A-94
 filtering error list A-98
 highlighting nets A-101, A-126
 loading error views A-95
 opening window A-95
 saving copies of the editor DRC error view
 A-104
 saving error view updates A-103
 saving errors in a report file A-102
 selecting errors in layout A-100
 viewing and fixing errors A-96
 viewing in area A-102
Error Selection tool A-100
estimated resistance A-124

exceptions for clock tree synthesis
 don't buffer net 5-28
 don't size cell 5-28
 don't touch subtree 5-27, 5-80
 exclude pins 5-11, 5-22
 float pin 5-23
 nonstop pins 5-22
 preserve hierarchy 5-29
 size-only cells 5-29
 stop pins 5-11, 5-27

exclusive move bounds, always-on synthesis
 12-21

executing
 command at startup 2-2
 script at startup 2-2

exiting IC Compiler 2-15

Expand Browser toolbar 5-149

expand_objects command A-111

expanding
 objects A-111

expansion commands A-111

exploring virtual hierarchy of flat design A-46

exporting
 GDSII 3-64
 OASIS 3-64
 parasitic data 3-64

extent 3-24

extract_rc command 6-146

extract_rp_group command 9-101

extracted timing model 5-6

extraction
 correlation with StarRC 3-51
 options 6-147
 postroute 6-146

F

files, bytecode-compiled 2-9

filler cell
 inserting pad fillers 7-56
 inserting well fillers 7-55

removing pad fillers 7-56
removing standard cell fillers 7-53
removing standard cell fillers that have
 routing design rule violations 7-53
removing tap cells 7-6
removing well fillers 7-56

filler cells, inserting 7-49

fix objects A-118

fix objects for edit A-115

fixed cells
 legalization 9-53
 movement 4-80

Flat Design Hierarchy Browser dialog box A-46

flat design, exploring virtual hierarchy A-46

flatten_clock_gating command 5-108

flip_objects command A-112

flipping objects A-112

floorplan file, reading 3-21

floorplan objects, create object tools A-114

Flyline Settings panel A-71

flylines
 Analysis toolbar A-70
 displaying A-71
 Flyline Settings panel A-71

Follow Selection command A-40

following selected objects in graphic views
 A-40

G

GDSII format 3-64

generated clocks
 as clock tree roots 5-8

get_always_on_logic command 12-41

get_bounds command 4-16

get_cts_scenario command 5-91

get_drc_errors command 6-134

get_license command 2-10

get_magnet_cells command 4-83

get_placement_blockages command 4-5, 4-10

get_route_guides command 6-33
get_route_opt_zrt_crosstalk_options command 6-67
get_routing_blockages command 6-37
get_selection command A-13
get_timing_paths command A-86
global routing
 description 6-80
 layer-based congestion map 6-114
 options
 descriptions 6-54
 reporting 6-54
 setting 6-54
 running 6-80
global skew
 multicorner-multimode 5-53
global skew, defined 5-53
Go Back in Zoom and Pan History command A-40
Go Forward in Zoom and Pan History command A-40
Graphical User Interface (GUI) 1-3
grid snapping A-115
grid spacing A-58
grids
 displaying or hiding A-58
 lithography grid A-58
 user grid A-58
groups
 creating plan groups A-114
 relative placement, defined 9-10
guard bands, voltage area 12-18
GUI 1-3
 closing A-2, A-23
 displaying on named terminal 2-2
 help starting 2-2
 hiding A-2, A-23
 opening A-2
 starting while executing command 2-2
 starting while executing script 2-2
GUI preferences
saving A-22
setting A-22
setting GUI command logging preferences A-23
GUI windows A-2, A-4
 menu bar A-5
 status bar A-6
 toolbars A-5
 view windows A-6
gui_add_missing_vias command A-113
gui_check_drc_errors command A-129, A-130
gui_get_layer_widths command A-123
gui_mouse_tool command A-108
gui_set_current_task command A-21
gui_set_flat_hierarchy_color command A-49
gui_set_layer_widths command A-123
gui_set_mouse_tool_option command A-108
gui_show_form command A-10
gui_start command A-2
gui_stop command A-23
gui_trim_dangling_wires command A-113
gui_unset_flat_hierarchy_color command A-49
gui_write_layout_image command A-56
guides
 creating pin guides A-115
 creating route guides A-114

H

help command 2-5
Help system
 accessing A-18
 searching for text in A-19
Hercules
 design rule checking
 distributed 6-126
 environment, setting up 6-123
 layer mapping file 6-125
 hiding the GUI A-2, A-23

hierarchy browser
exploring design hierarchy A-44
logic hierarchy view A-44

high-fanout nets, buffering 4-24

high-fanout synthesis
during place_opt 4-24
standalone 4-24
using compile_clock_tree 5-100
voltage area support 12-34

high-level design flow, multivoltage designs 12-6

Highlight tool A-26, A-33

Highlight toolbar A-34

highlighting
critical path A-35
selected objects A-35
timing paths A-35

histograms A-25, A-87
clock arrival histograms 5-148
endpoint slack histogram A-87
net capacitance histogram A-88
path slack histogram A-87

history
command log A-16
view A-15

hotkeys report, displaying A-12

|

IC Compiler 2-2
exiting 2-15
icc_shell commands 2-4
starting 2-2

IC Compiler packages
IC Compiler 1-2
IC Compiler-DP 1-2
IC Compiler-PC 1-2
IC Compiler-XP 1-2

icc_output.txt file 2-3

icc_shell command 2-3
listing startup options 2-2

starting IC Compiler 2-2
using 2-4

icc_shell command-line interface 1-3
icc_shell> prompt 2-3, 2-4

ignore attribute, removing 9-91

image
capturing window or view A-55

importing
GDSII 3-65
OASIS 3-65

importing, Verilog 3-14, 3-15

In-Design Rail Analysis
electromigration analysis 11-39
inrush current analysis 11-43
power network integrity analysis 11-18
voltage drop analysis 11-32

In-Design rail analysis
library requirements 11-7

information
displaying for objects in graphic views A-31
displaying InfoTips in layout views A-28

InfoTips A-28

insert_pad_filler command 7-56

insert_tap_cells_by_rules command 7-5

insert_well_filler command 7-55

inserting enable-type level shifters,
multivoltage designs 12-28

inserting isolation cells, multivoltage designs 12-28

inspecting timing paths A-92

interactive clock tree synthesis 5-164

interactive clock tree synthesis window,
displaying 5-147

intercell spacing rules, defining 4-17, 4-18

interclock delay balancing
clock group, specifying 5-39
multicorner-multimode 5-39
running 5-103
timing analysis, router for 5-103

interclock skew, balancing 5-39

interface logic model
 logic-level balancing in clock tree synthesis
 5-64
interfaces
 GUI 1-3
 icc_shell 1-3
is_isolation_cell attribute 12-27, 12-28
is_level_shifter attribute 12-27, 12-28
isolation cell 3-24
isolation cells
 and level shifters, multivoltage designs 12-5
 inserting 12-28
isolation_cell_enable_pin attribute 12-28

J

jog wires, preventing wrong way 6-30

K

keepout margins
 cell-specific, setting 4-4
keyboard shortcuts A-12

L

layer constraints
 defining 3-52
layout editing 6-135, 6-137, 6-138, 6-139,
 6-140, 9-73
layout editing tools 6-135, 6-140, A-105
 Create Custom Wire tool 6-144
layout tools, analysis A-70
layout view A-24, A-56
 refreshing A-41
 save screenshot of A-55
layout window A-19
legalize_placement command 12-27
level shifter 3-24

level shifter commands, multivoltage designs
 12-22
level shifters
 and isolation cells, multivoltage designs 12-5
 placing 12-27
level_shifter_enable_pin attribute 12-28
library data, preparing 1-5
library groups (scaling) 3-46
license queuing
 enabling 2-10
 environment variables 2-10
 SNPS_MAX_QUEUETIME 2-11
 SNPS_MAX_WAITTIME 2-11
 SNPSLMD_QUEUE 2-10
license_users command 2-10
licenses
 checking out 2-10
 enabling queuing 2-10
 listing 2-10
 releasing 2-11
 using 2-9
 working with 2-9
link_library variable 3-3
linking the design 3-4
Linux shell 2-3
list_drc_error_types command 6-134
listing commands A-12
listing startup options 2-2
lithography grid, displaying or hiding A-58
locking objects for edit A-115
log file
 record and replay GUI session A-23
 using 2-3
log interactive operation A-108
log view A-13
logic hierarchy view A-44
logic libraries 1-5
low-effort flow 1-7
lwindow
 save image of A-55

M

- macros
 - voltage area 12-20
- magnet placement 4-80
- magnet_placement command 4-80
- main library, defined 3-3
- main window A-8
- man page viewer A-17
- map file for TLUPlus 3-40
- Map Mode panel 7-38, A-75
- map modes
 - Analysis toolbar A-70
 - displaying A-75
 - Map Mode panel A-75
- map_retention_cell command 12-29
- maximum timing information 3-44
- menu bar A-5
- menu commands
 - choosing A-9
 - finding A-10
- menus
 - choosing current task A-21
 - keyboard shortcuts A-12
- metal fill extraction
 - emulation 7-57
 - real 7-57
- metal fill, inserting 7-57
- methodology 1-4, 6-4
- Milkyway database
 - about 3-3
 - saving a design 3-60
- Milkyway design library
 - changing settings 3-10
 - creating 3-5
 - defined 3-4
 - reporting
 - Milkyway reference libraries 3-7
 - units 3-9, 8-15
- Milkyway format, saving in 1-6
- Milkyway reference library
- changing 3-10
- defined 3-4
- minimum insertion delay, defined 5-53
- minimum timing information 3-44
- missing vias, adding A-113
- Mouse Tool Options toolbar A-28, A-30, A-107
- mouse tools A-109
 - Advanced Route tool A-112
 - Align/Distribute tool A-110, A-111
 - Area Push tool A-113
 - Copy tool A-109
 - Create Net tool A-113
 - Create RDL Route tool A-112
 - Create Route tool A-112
 - Create Routing Corridor tool A-113
 - Create Terminal tool A-114
 - Create Text tool A-114
 - Create Via tool A-113
 - Custom Wire tool A-114
 - Delete tool A-110
 - Edit tool (see Move/Resize tool)
 - Highlight tool A-26, A-33
 - Mouse Tools toolbar A-26
 - Move Pins on Edge tool A-111
 - Move/Resize tool
 - move objects A-109
 - resize objects A-109
 - Pan tool A-26
 - Query tool A-26, A-31
 - Reshape tool A-110
 - Ruler tool A-26, A-58
 - Selection tool A-26, A-29
 - Split tool A-110
 - Spread Wire tool A-113
 - Stretch Wire tool A-110
 - Window Stretch tool A-110
 - Zoom In tool A-26
 - Zoom Out tool A-26
- Mouse Tools toolbar A-26
- move bound
 - deleting 4-16

reporting 4-16
 move pins on edge tool A-111
 move_objects command A-109
 move_pins_on_edge command A-111
 Move/Resize tool
 moving objects A-109
 resizing objects A-109
 moving and stretching objects A-110
 moving objects A-109
 Area Push tool A-113
 moving pins A-111
 multicorner-multimode
 all_scenarios command 3-38
 scenario reduction 3-36
 set_min_library command 3-44
 multiple instances, support for 3-15
 multithreading
 defined 2-12, 2-14, 6-42
 multivoltage designs
 always-on power wells 12-21
 always-on synthesis 12-40
 creating voltage areas 12-16
 high-level design flow 12-6
 inserting enable-type level shifters 12-28
 inserting isolation cells 12-28
 level shifter commands 12-22
 level shifter placement 12-27
 level shifters and isolation cells 12-5
 logical power and ground connections 3-28
 nested voltage areas 12-19
 placing and optimizing 12-40
 power domains 12-3
 power domains and voltage areas 12-4
 voltage areas 12-4

N

Named Zoom and Pan Settings command
 A-40
 nested voltage areas, multivoltage designs
 12-19

net capacitance histogram
 histogram views A-88
 Timing Views toolbar A-87
 net connectivity
 Analysis toolbar A-70
 Connectivity Settings panel A-72
 displaying A-72
 net layer constraints, specifying 6-13
 net shapes, Create Net tool A-113
 net shielding 7-41
 net switching information, loading 8-15
 net, report estimated resistance A-124
 net, supply 3-24
 noise modeling 8-8
 nondefault routing rules
 applying 6-24
 defining 6-16
 reporting 6-25, 6-26

O

OASIS format 3-64
 object alignment tool A-110
 object copying tool A-109
 object distribution tool A-110
 object expansion commands A-111
 object flipping commands A-112
 object information
 InfoTips A-28
 Query tool A-31
 object moving tools
 MoveResize A-109
 Stretch Wire A-110
 Window Stretch A-110
 object properties A-42
 object removing tool A-110
 object reshaping tool A-110
 object resizing tools
 MoveResize A-109
 Stretch Wire A-110

Window Stretch A-110
object rotation commands A-111
object snapping options for layout editing A-118
object splitting tool A-110
object spread tool A-111
object stretching tool A-110
objects
 create floorplan object tools A-114
 fix or unfix for editing A-118
 select by name A-51
 selecting in graphic views A-29
on-chip-variation, clock tree clustering 5-62
online Help
 accessing A-18
 searching for text in A-19
open_mw_cel command A-133
opening the GUI A-2
operating condition analysis modes
 best-case and worst-case 3-43
 on-chip variation 3-43
operating conditions 3-42
optimization feasibility analysis 5-143
optimize_clock_tree command 5-101
optimize_pre_cts_power command 5-36
order_rp_groups command 9-102
orientations A-112
orientations, displaying or hiding for cells A-60
orienting cells A-112
overlapping clock path, defined 5-7
overlapping objects A-28
overlays, layout view A-66
overriding DISPLAY value 2-2
Overview panel A-57

P

packages, IC Compiler 1-2
pad filler
 inserting 7-56

removing 7-56
Pan to Selection command A-37
Pan tool A-26
panels
 Connectivity Settings panel A-72
 Edit Snapping Settings panel A-118
 Flyline Settings panel A-71
 Map Mode panel A-75
 Overview panel A-57
 Query panel A-31
 Timing Analysis Driver panel A-86
 View Settings panel
 layout views A-61
 schematic views A-91
 Visual Mode panel A-72
path data element table, path inspector window A-92
path inspector
 panels A-92
 path schematic A-92
 viewing path profiles A-89
path inspector windows A-92
path profile views
 timing analysis A-89
 Timing Views toolbar A-87
path profiles
 in path inspector A-92
 path inspector window A-92
path schematics A-89
 in path inspector A-92
path slack histogram
 histogram views A-87
 Timing Views toolbar A-87
path specifications for TLUPlus 3-41
path summaries, path inspector window A-92
paths
 select A-53
physical bus
 creating
 automatic 6-138
 manual 6-137

modifying 6-139
routing 6-140

physical bus routing tool A-112

physical bus, modifying 6-139

physical bus, routing 6-140

physical data
copying 3-21
validating 3-25

physical datapath, defined 9-1

physical optimization, voltage area support 12-35

physical signoff options
reporting 6-124
setting 6-124

physical synthesis
placing and optimizing designs 12-40
voltage-area-aware capabilities 12-34

physopt_check_site_array_overlap variable 12-15

physopt_rp_enable_orient_opt variable 9-22

pin guides
creating objects A-115

pin tapering, controlling 6-40

pin-to-pin estimated resistance report A-124

place_opt command 4-29, 12-4, 12-5, 12-27, 12-34

place_opt_feasibility command 4-47

placement
categorized timing report 4-67
filters 4-68
magnet 4-80
multicorner-multimode 4-29
refining 4-90
subgroups 4-69

placement and optimization 1-5
checkpointing 4-45
customizing 4-29
feasibility 4-47
multivoltage 12-40
physical synthesis 12-40
skip initial placement 4-39

Synopsys physical guidance 4-39
placement area utilization 4-56
placement blockages
creating 4-5
creating objects A-114
thin channels 4-10

placement statistics 6-114

plan groups
creating objects A-114

pop operation, edit-in-place A-133

port, supply 3-24

postroute extraction 6-146

power and ground net associations, voltage area support 12-39

power and ground network
analyzing (see In-Design Rail Analysis)
creating logical connections
multivoltage designs 3-28
single-voltage designs 3-27
reporting logical connections 3-26

power domain 3-24
extent 3-24
scope 3-24

power domains
multivoltage designs 12-3
voltage areas 12-4

power state table 3-24

power switch 3-24

powerPrecision, in the technology file 11-7

preferences
saving A-22
setting A-22
setting GUI command logging preferences A-23

preferred routing direction
reporting 6-26
setting 6-26

prerequisites, routing 6-5

preroute scaling factors, setting 3-52, 3-53, 3-54

preroute_focal_opt command 4-86

preview selection A-28
previewing commands in dialog boxes A-12
process scaling factors 6-148
process_particle_probability_file command 7-37
propagate_constraints command 12-39
properties, viewing and editing for selected object A-42
psynopt command 12-4
push operation, edit-in-place A-133
pushing objects A-113
PVT operating conditions 3-42
PVT scaling 3-46

Q

Query panel A-31
Query tool A-26, A-31
querying objects A-31
queuing licenses 2-10
quick flow 1-7
quitting IC Compiler 2-15

R

RC calculations, preparing 3-40
RC estimation
 coefficients, specifying 3-53
 scaling factors, setting 3-52
read_def command 3-20
read_drc_error_file command 6-133
read_floorplan command 3-21
read_parasitics command 3-54
read_sdc command 3-48
read_sdf command 3-54
read_stream command 3-65
read_verilog command 3-14, 3-15
reading
 DEF file 3-20

Verilog 3-14, 3-15
reapply edits A-115
reapplying edits in layout view A-116
recording GUI session A-23
redo command A-115, A-116, A-117
redo edits A-116
redrawing the active view A-41
reference control file
 writing 3-10
reference libraries, Milkyway
 reporting 3-7
 setting 3-6
reference, clock tree
 specifying 5-43
refine_placement command 4-90
refreshing the active view A-41
relative coordinates A-28
relative placement
 aligning by pins 9-23
 anchoring at a specified location 9-13
 cell placement using bounds 9-55
 changing specific items within a group 9-99
 commands 9-103
 compression 9-44
 considerations for using 9-9
 example script 9-7
 flow 9-5
 group
 adding items to 9-21
 adding or removing items from in GUI A-81
 creating 9-10
 defined 9-10
 extracting 9-101
 ordering extracted 9-102
 renaming 9-98
 hierarchical group
 adding 9-35
 exploring in GUI A-81
 for instantiation 9-38
 included, defined 9-35
 instantiated, defined 9-35

syntax to include 9-36
syntax to instantiate 9-38
hierarchy view A-81
in a design containing obstructions 9-51, 9-52
incremental relative placement 9-58
keepout
 adding 9-49
 adding or removing in GUI A-81
leaf cell
 adding or removing in GUI A-81
 specifying orientation 9-22
limitations 9-104
methodology 9-6
modifying a group 9-93
modifying information 9-93
orientation optimization
 defined 9-22
 directing 9-22
positions for data 9-10
preserving relative placement structures 9-62
relative placement information, ignoring 9-91
removing 9-92
removing items from a group 9-99
required inputs 9-9
straddling 9-43
ungrouping 9-46
uniquifying 9-48
voltage area support 12-40
writing information to script file 9-90
relative placement groups
 exploring in GUI A-81
 relative placement hierarchy view A-81
 removing in GUI A-81
 removing items from in GUI A-81
relative placement hierarchy view A-81
relative placement view window
 editing 9-80, 9-81
 viewing net connections 9-77
remove_all_spacing_rules command 4-18
remove_annotations command 3-54
remove_bounds command 4-16
remove_buffer_tree command 4-24
remove_clock_tree command 5-80
remove_clock_tree_exceptions command
 -dont_buffer_nets 5-28
 -dont_size_cells 5-28
 -dont_touch_subtrees 5-27
 -exclude_pins 5-22
 -float_pins 5-25
 -non_stop_pins 5-22
 -preserve_hierarchy 5-30
 -size_only_cells 5-29
 -stop_pins 5-27
remove_filler_withViolation command 7-53
remove_from_rp_group command 9-99
remove_ideal_network command 3-48
remove_keepout_margin command 4-5
remove_level_shifters command 12-27
remove_license command 2-11
remove_net_routing_layer_constraints command 3-52
remove_objects command A-110
remove_placement_blockage command 4-10
remove_preferred_routing_direction command 6-27
remove_route_guide command 6-33, 6-37
remove_routing_blockage command 6-37
remove_rp_group_options command 9-91
remove_rp_groups command 9-92
remove_sdc command 3-48
remove_self_gating_logic command 5-32
remove_skew_group command 5-161
remove_stdcell_filler command
 -pad 7-56
 -stdcell 7-53
 -tap 7-6
remove_target_library_subset command 4-20
remove_voltage_area command 12-21
remove_well_filler command 7-56
removing objects A-110

replay interactive operation A-108
replaying GUI session A-23
report estimated resistance A-124
report views A-54
report_bounds command 4-16
report_buffer_tree command 4-24
report_cell command 12-44
report_clock_timing command 5-139
report_clock_tree command 5-135
report_clock_tree_power command 5-32
report_constraint command 5-167
report_critical_area command 7-36
report_design command 6-114
report_drc_error_type command 6-134
report_error_coordinates command 6-10
report_inter_clock_delay_options command 5-43
report_keepout_margin command 4-5
report_milkyway_version 3-9
report_mw_lib command 3-7
report_net_routing_layer_constraints command 3-52
report_net_routing_rules command 6-25, 6-26
report_noise_calculation command 8-11
report_operating_conditions command 12-45
report_physical_signoff_options command 6-124
report_placement_utilization command 4-56
report_power_guide command 12-21
report_preferred_routing_direction command 6-26
report_preroute_focal_opt_strategy command 4-89
report_route_opt_strategy command 6-94
report_route_opt_zrt_crosstalk_options command 6-67
report_routing_rules command 5-54
report_scan_chain command 4-79
report_signal_em command 8-17
report_signal_em_calculation command 8-20
report_spacing_rules command 4-18
report_target_library_subset command 4-20, 12-45
report_timing command 4-58, 5-167
report_timing_derate command 3-46
report_tlu_plus_files command 3-42
report_units command 3-9, 8-15
report_voltage_area command 12-21, 12-44
report, clock tree
generating 5-135
types 5-136
reporting
cell placement statistics 6-114
routing and optimization strategy 6-94
routing statistics 6-114
reports
saving A-54
viewing A-54
reset_design command 3-48
reset_inter_clock_delay_options commands 5-42
reset_timing_derate command 3-46
Reshape tool A-110
resistance estimation A-124
resize_objects command A-109
resizing objects A-109
retention register 3-25
reusing commands A-16
reverse edits A-115
reversing edits in layout view A-116
rotate object commands A-111
rotate_objects command A-111, A-112
rotating objects A-111
routability, checking 6-8
route guide
creating 6-28
creating objects A-114
finding 6-33
preferred-direction-only 6-30

removing 6-33, 6-37
switch-preferred-direction 6-30

route_htree command 5-115

route_mesh_net command 5-112

route_opt command 12-34

route_zrt_eco command 6-110

route_zrt_group command 6-72, 6-74

routing

- automatic 6-91
- clock nets 6-72
- critical nets 6-74
- ECO 6-110
- interactive
 - multiple nets 6-140
 - physical bus 6-140
 - single net 6-135
- over macros 6-30
- postroute extraction 6-146
- prerequisites 6-5
- route_opt 6-93
- running individual steps 6-80
- signal nets 6-79
- special nets 6-72, 6-74
- verifying
 - using IC Compiler 6-120

routing and optimization strategy

- reporting 6-94
- setting 6-93

routing blockage

- creating 6-36
- creating objects A-114
- finding 6-37
- removing 6-37

routing corridors

- Create Routing Corridor tool A-113

routing direction, setting preferred 6-26

routing layers

- specifying 6-13

routing layers, clock tree

- default 5-58
- setting 5-58

routing rule

- clock tree
 - default 5-54
 - nondefault 5-54
- defining 5-54
- nondefault, defined 5-54
- reporting 5-54

routing rules (nondefault), defining 6-16

routing statistics 6-114

routing tools

- Advanced Route A-112
- Create RDL Route A-112
- Create Route A-112
- Custom Wire A-114

routing tracks, creating objects A-114

rp_group_inclusions command 9-89

rp_group_instantiations command 9-89

rp_group_references command 9-90

Ruler tool A-26, A-58

S

save window or view image A-55

save_mw_cel command 3-60, A-103, A-135

saving

- design using Milkyway 3-60
- designs 1-6
- log messages A-14
- preferences A-22
- reports A-54
- transcript of session commands A-13, A-16

SBPF

- reading 3-54
- writing 3-64

scaling factors

- process 6-148

scaling factors, preroute, setting 3-52, 3-53, 3-54

scaling library groups 3-46

scan chains

- partitioning

vertical 4-34
reporting 4-79
viewing 4-79
scenario reduction, multicorner-multimode 3-36
schematic view A-25
Schematic View panel, path inspector window A-92
schematic views
 design schematics A-90
 path schematics A-89
 refreshing A-41
 symbol view A-90
Schematics toolbar A-90
scope 3-24
Screenshot of layout view A-55
scripts
 executing at startup 2-2
SDC
 reading 3-48
 writing 3-64
SDF files, reading 3-54
search and repair, running 6-86
search_path variable 3-3, 3-41
searching the transcript A-14
select by name dialog box A-51
Select By Name tool A-36
Select By Name toolbar A-36
select timing paths A-53
selected objects, following in graphic views A-40
selecting objects in graphic views A-29
selecting only highlighted objects A-36
selection preview A-28
Selection tool A-26, A-29
session log A-13
set_ahfs_options command 4-24
set_checkpoint_strategy command 4-45
set_clock_tree_exceptions command
-dont_buffer_nets 5-28
-dont_size_cells 5-28
-dont_touch_subtrees 5-27
-exclude_pins 5-22
float pin options 5-23
-non_stop_pins 5-22
-preserve_hierarchy 5-29
-size_only_cells 5-29
-stop_pins 5-27
set_clock_tree_options command 6-24
 -layer_list 5-58
 -max_capacitance 5-50
 -max_fanout 5-50
 -max_transition 5-50
 -routing_rule 5-54
 -target_early_delay 5-53
 -target_skew 5-53
 -use_default_routing_for_sinks 5-54
set_clock_tree_references command 5-43
set_congestion_options command 4-12
set_delay_calculation_options 3-49
set_delay_estimation_options command 3-51, 3-52, 3-53, 3-54
set_em_options command 8-18
set_extraction_options command 6-147
set_gui_stroke_binding command A-69
set_gui_stroke_preferences command A-69
set_ignored_layers command 6-13
set_inter_clock_delay_options command 5-40
set_isolation command 12-28
set_keepout_margin command 4-4
set_left_right_filler_rule command 7-52
set_level_shifter_strategy command 12-23
set_level_shifter_threshold command 12-23, 12-25, 12-26
set_lib_cell_spacing_label command 4-17, 4-18
set_max_transition command 5-52
set_min_library command 3-44
set_mw_lib_reference command 3-10

set_net_routing_layer_constraints command
 3-52

set_net_routing_rule command 6-25

set_object_fixed_edit command A-115

set_object_snap_type command A-115,
 A-118, A-119, A-120

set_operating_conditions command 3-55,
 12-24

set_optimize_dft_options command 4-34

set_physical_signoff_options command 6-124,
 7-58

set_power_guide command 12-21

set_power_net_to_voltage_area command
 12-39

set_preferred_routing_direction command
 6-26

set_preroute_focal_opt_strategy command
 4-89

set_route_opt_strategy command 6-93

set_route_opt_zrt_crosstalk_options
 command 6-66

set_rp_group_options command 9-93

set_si_options command 3-49, A-83

set_skew_group command 5-160

set_spacing_label_rule command 4-17, 4-18

set_target_library_subset command 4-18

set_timing_derate command 3-46

set_tlu_plus_files command 3-41, 3-42, 7-57

set_write_stream_options command 3-64

setting preferences A-22

setting the current design 3-17

sh_command_log_file variable 2-3

shape splitting tool A-110

shapes, Create Net tool A-113

shell command line
 entering commands 2-4

shell commands 2-4

shell interface 1-3

shielding nets 7-41

shortcut keys A-12

signal electromigration
 performing analysis 8-17

signal integrity
 crosstalk prevention 6-66
 crosstalk reduction 6-65
 defined 8-1, 8-2

options
 commands affected 8-3
 setting 8-3

single net, routing 6-135

size_only attribute 12-24, 12-25, 12-28

skew
 balancing between clocks 5-39
 global 5-53

skew group
 definition 5-159
 guidelines 5-159

snapping options for layout editing A-118

snapping, enable or disable A-115

snps_derived attribute 12-49

SolvNet
 accessing xxxvi
 documentation xxxiv

SPEF (Standard Parasitic Exchange Format)
 files
 reading 3-54
 writing 3-64

Split tool A-110

split_clock_gates command 5-37

split_clock_net command 5-109

split_objects command A-110

spread objects tool A-111

Spread Wire tool A-113

spread_zrt_wires command 7-39

spreading objects A-111

spreading wires A-113

standard cell filler
 removing 7-53
 removing fillers that have routing rule
 violations 7-53

rules, defining 7-52

standard cell filler, inserting 7-49
standard cell row, adding end caps 7-53
starting IC Compiler 2-2
 executing commands 2-2
 executing scripts 2-2
starting `icc_shell` 2-2
starting the GUI 2-2
 listing startup options 2-2
startup options, -help list 2-2
static noise
 defined 8-2
 reporting 8-11
 violations 8-11
 visual mode 8-11
status bar A-6, A-28
Status panel, path inspector window A-92
straddling, in relative placement, defined 9-43
Stretch Wire tool A-110
stretching and moving objects A-110
stretching wires A-110
stroke activated commands A-68
stroke commands A-68
supply net 3-24
supply port 3-24
switch, power 3-24
symbol views A-90

used by IC Compiler 1-3
TclPro
 bytecode-compiled files 2-9
 limitations 2-9
 procomp 2-9
technology file
 specifying 3-5
technology (.tf) file
 defined 3-4
 writing 3-10
terminals, Create Terminal tool A-114
text, Create Text tool A-114
tie cell insertion 4-21
timing analysis
 histograms A-87
 path inspector windows A-92
 preparing 3-40
 timing analysis driver A-86
 timing analysis window A-85
 Timing Views toolbar A-87
timing analysis driver A-86
Timing Analysis Driver panel A-86
timing constraints
 removing 3-48
 setting 3-48
timing goals, clock tree 5-53
timing information
 maximum 3-44
 minimum 3-44
timing libraries 1-5
 derating factors 3-46
timing paths
 analyzing A-85
 endpoint slack histogram A-87
 path slack histogram A-87
 selecting A-53
 timing analysis driver A-86
 viewing in path inspector A-92
timing report 4-58, 5-167
 crosstalk analysis 8-10
Timing Views toolbar A-87

T

Tab panel, path inspector window A-92
tap cell
 inserting 7-5
 removing 7-6
tap cell array
 adding 7-3
target library
 subsetting 4-18
target_library variable 3-4, 4-20
task, layout window A-21
Tcl (Tool Command Language)

- timing_enable_multiple_clocks_per_reg variable 5-12
- TLUPlus
 - map file 3-40
 - models 3-40
 - reporting files 3-42
 - specifying paths 3-41
 - validating files 3-42
- toggle snapping A-115
- Tool Command Language (Tcl)
 - used by IC Compiler 1-3
- tool command language (Tcl) mode 1-3
- toolbars A-5
 - Analysis toolbar A-70
 - Cells List 3-17
 - console A-12
 - Edit In Place A-135
 - Expand Browser 5-149
 - Highlight toolbar A-34
 - Mouse Tool Options toolbar A-28, A-30, A-107
 - Mouse Tools toolbar A-26
 - Schematics A-90
 - Select By Name toolbar A-36
 - Timing Views A-87
 - Undo A-116
 - View Zoom/Pan toolbar A-37
 - Zoom and Pan History toolbar A-40
- tools A-112, A-114
 - Align/Distribute tool A-110, A-111
 - Area Push tool A-113
 - Copy tool A-109
 - create floorplan object tools A-114
 - Create Net tool A-113
 - Create RDL Route tool A-112
 - Create Route tool A-112
 - Create Routing Corridor tool A-113
 - Create Terminal tool A-114
 - Create Text tool A-114
 - Create Via tool A-113
 - Delete tool A-110
 - Edit tool (see Move/Resize tool)
- Highlight tool A-26, A-33
- layout editing tools A-105
- left mouse button tools A-26
- Move Pins on Edge tool A-111
- Move/Resize tool
 - move objects A-109
 - resize objects A-109
- Pan tool A-26
- Query tool A-26, A-31
- Reshape tool A-110
- Ruler tool A-26, A-58
- Select By Name tool A-36
- Selection tool A-26, A-29
- Split tool A-110
- Spread Wire tool A-113
- Stretch Wire tool A-110
- Window Stretch tool A-110
- Zoom In tool A-26
- Zoom Out tool A-26
- top-level windows A-4
- track assignment
 - description 6-85
 - options
 - descriptions 6-57
 - reporting 6-57
 - setting 6-57
 - running 6-85
- transcript of session commands, saving A-13
- trim dangling wire ends A-113

U

- uncommit_fp_soft_macros command 9-57
- undo command A-115, A-116, A-117
- undo edits A-116
- Undo toolbar A-116
- unfix objects A-118
- unfix objects for edit A-115
- uniquify_fp_mw_cel command 3-15
- unitCurrentName, in the technology file 11-7
- unitPowerName, in the technology file 11-7

units
 default 3-4
 saving with design 3-60
 saving with SDC 3-64

unitVoltageName, in the technology file 11-7

unitVoltagePrecision, in the technology file 11-7

UNIX
 DISPLAY environment variable A-2
 shell 2-3

unlocking objects for edit A-115

unset_power_guide command 12-21

update_clock_latency command 5-104

UPF commands
 map_retention_cell 12-29
 set_isolation 12-28

UPF strategy
 isolation 12-28

user grid, displaying or hiding A-58

user interfaces 1-3

user shapes, Create Net tool A-113

utilization 4-56

V

variables
 cts_do_characterization 5-135
 cts_dont_touch_subtrees 5-27
 cts_enable_rc_constraints 5-74
 cts_fix_clock_tree_sinks 5-88
 cts_move_clock_gate 5-87
 cts_rc_relax_factor 5-74
 cts_self_gating_connect_scan_enable 5-32
 cts_self_gating_data_toggle_rate_filter 5-32
 cts_self_gating_num_regs 5-32
 cts_self_gating_num_regs_max_percent 5-32
 cts_self_gating_num_regs_max_percent_to_tal_cell 5-32
 cts_self_gating_reg_power_filter 5-33
 cts_self_gating_wns_backoff 5-33

cts_use_debug_mode 5-135
 cts_use_lib_max_fanout 5-51
 cts_use_sdc_max_fanout 5-51

DISPLAY A-2

link_library 3-3

physopt_check_site_array_overlap 12-15

physopt_rp_enable_orient_opt 9-22

placer_max_cell_density_threshold 4-12

search_path 3-3, 3-41

sh_command_log_file 2-3

SNPS_MAX_QUEUEUTIME 2-11

SNPS_MAX_WAITTIME 2-11

SNPSLMD_QUEUE 2-10

target_library 3-4, 4-20

tieoff_hierarchy_opt 4-21

tieoff_hierarchy_opt_keep_driver 4-21

timing_enable_multiple_clocks_per_reg 5-12

verify_route command 6-120

Verilog
 reading 3-14, 3-15
 writing 3-63

via arrays, Create Via tool A-113

via resistance
 scaling 3-53

vias
 Create Via tool A-113

vias, add missing A-113

View Settings panel
 design overlays on layout view A-66
 layer properties in layout view A-62
 object properties in layout view A-62
 object properties in schematic view A-91
 save or load settings A-64
 using with active layout view A-61

view window
 save image of A-55

view windows A-6

View Zoom/Pan toolbar A-37

viewing
 man pages A-17

- object properties A-42
- reports A-54
- scan chains 4-79
- views
 - clock graph A-25
 - clock tree fanout browser 5-148
 - design schematics A-90
 - histogram A-25
 - histograms A-87
 - layout A-24, A-56
 - path inspector window A-92
 - path profiles A-89
 - path schematics A-89
 - schematic A-25
 - symbol A-90
- virtual flat placement 9-55
- virtual hierarchical cells A-46
- virtual hierarchy routing, voltage area support 12-35
- virtual hierarchy, Flat Design Hierarchy Browser dialog box A-46
- Visual Mode panel A-72
- visual modes
 - Analysis toolbar A-70
 - delta delay 8-10
 - displaying A-72
 - static noise 8-11
 - Visual Mode panel A-72
- voltage and temperature scaling 3-46
- voltage area 3-24
 - defined 6-69
 - routing behavior
 - controlling for a net 6-70
 - controlling for the design 6-70
 - default 6-69
- voltage areas
 - creating 12-16
 - creating for multivoltage designs 12-16
 - creating objects A-115
 - guard bands, using 12-18
 - macro cells 12-20
- multivoltage designs 12-4
- power domains 12-4
- power domains, associating 12-16
- removing 12-21
- reporting 12-21
- updating 12-17
- voltage-area-aware capabilities 12-34
 - automatic high-fanout synthesis 12-34
 - clock tree synthesis and optimization 12-36
 - physical optimization 12-35
 - power and ground nets, associating 12-39
 - using relative placement 12-40
 - virtual hierarchy routing 12-35
 - voltage-area-based optimization 12-35
 - voltage-area-based optimization 12-35
- well filler
 - inserting 7-55
 - removing 7-56
- widen_zrt_wires command 7-40
- Window Stretch tool A-110
- windows
 - layout A-19
 - main A-8
 - menu bar A-5
 - status bar A-6
 - timing analysis A-85
 - toolbars A-5
 - top-level A-4
 - view windows A-6
 - working environment A-2
- wire routing tools
 - Advanced Route A-112
 - Create RDL Route A-112
 - Create Route A-112
 - Custom Wire A-114
- wire spreading 7-39
- wire stretching tool A-110
- wire widening 7-40

wire, report estimated resistance A-124
wires
 creating with layout editor 6-144
wires, spreading A-113
wires, trim dangling wire ends A-113
working environment A-2
write_def command 3-63
write_mw_lib_files command 3-10
write_parasitics command 3-64, 6-147
write_rp_groups command 9-90
write_sdc command 3-64
write_stream command 3-64
writing
 DEF file 3-63
 Verilog 3-63

X

XOR self-gating 5-30

Y

yield, optimizing

displaying critical area heat maps 7-38
performing wire spreading 7-39
performing wire widening 7-40

Z

Zoom and Pan History toolbar A-40
Zoom Fit All command A-37
Zoom Follows Selection command A-37
Zoom In 2x command A-37
Zoom In tool A-26
Zoom Out 2x command A-37
Zoom Out tool A-26
Zoom to Selection command A-37
Zroute
 features 6-3
 flow 6-4
Zroute options 6-43
 common route options 6-44
 detail route options 6-57
 global route options 6-54
 track assignment options 6-57