cadence®

Virtuoso Abstract Generator User Guide

Product Version ICADVM20.1
October 2020

© 2020 Cadence Design Systems, Inc. All rights reserved.
Printed in the United States of America.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Open SystemC, Open SystemC Initiative, OSCI, SystemC, and SystemC Initiative are trademarks or registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries and are used with permission.

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522. All other trademarks are the property of their respective holders.

Restricted Permission: This publication is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this publication, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used only in accordance with a written agreement between Cadence and its customer.
2. The publication may not be modified in any way.
3. Any authorized copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.
4. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Contents

Preface . . . 17
Scope . . . 18
Licensing Requirements . . . 19
Standalone Abstract Generator . . . 19
Integrated Abstract Generator . . . 21
Related Documentation . . . 22
What's New and KPNS . . . 22
Installation, Environment, and Infrastructure . . . 22
Technology Information . . . 22
Virtuoso Tools . . . 23
Relative Object Design and Inherited Connections . . . 23
Additional Learning Resources . . . 23
Video Library . . . 23
Virtuoso Videos Book . . . 23
Rapid Adoption Kits . . . 24
Help and Support Facilities . . . 24
Customer Support . . . 24
Feedback about Documentation . . . 25
Typographic and Syntax Conventions . . . 26

1
Overview of Abstract Generator . . . 27
Use Model for Abstract Generation . . . 29
Block Override Support in Abstract Generation . . . 29
Concept of Overriding Terminals and Instance Terminals . . . 30
Flows Supported by Abstract Generator . . . 32
GDSII to LEF . . . 32
DEF to OA . . . 32
LEF to OA . . . 33
Virtuoso Layout Flow . . . 34

© 2020 All Rights Reserved.

2 Standalone Abstract Generator . . . 37

Getting Started . . . 38
- Setting Up the Abstract Generator Environment . . . 38
- Launching the Standalone Abstract Generator . . . 39

Getting Familiar with the Interface . . . 42
- Main Window . . . 43
- Menu Bar . . . 44
- Toolbar . . . 46
- Bin Pane . . . 48
- Cell Pane . . . 51
- Log Window . . . 54
- Command Entry Field . . . 55
- Command History Window . . . 56
- Progress Window . . . 57

Customizing the Interface . . . 58
- Customizing Fonts . . . 58
- Customizing the Appearance . . . 60
- Applying the Font and Appearance Changes . . . 67
- Customizing the Abstract Generator Flow Steps . . . 68

Generating Abstracts . . . 73

Running Abstract Generator in Batch Mode . . . 78

Log, Replay, and Record File Behavior . . . 80
- Specifying the CDS.log File Name . . . 81
- Filtering Messages . . . 81

Saving and Loading the Options and Status Files . . . 83

Optimizing the Performance of Abstract Generator . . . 85

3 Pins Step . . . 89

Preparing for Abstract Generation . . . 90

Map Tab . . . 91
- Mapping Text Labels to Pins . . . 92
- Specifying Pin Names . . . 96

© 2020 All Rights Reserved.

Excluding Existing Terminals . . . 97
Excluding Existing Pins . . . 97
Text Tab . . . 98
Specifying the Search Depth . . . 98
Manipulating Text Labels . . . 99
Generating Additional Pins from Labels . . . 100
Preserving Text Labels from the Layout View . . . 100
Restricting Pins to the PR Boundary . . . 101
Boundary Tab . . . 102
Specifying the Creation of the PR Boundary . . . 102
Specifying the Layers for PR Boundary Calculation . . . 103
Manipulating the Dimensions of the PR Boundary . . . 104
Blocks Tab . . . 105
Preserving Local Blockages in Routed Blocks . . . 105
Creating Power Pins from Routing . . . 105
Creating N-Well and P-Well Pins . . . 107
N-Well Pins . . . 107
P-Well Pins . . . 108
Creating Multiple N-Well and P-Well Pins . . . 110

4
Extract Step . . . 113

Specifying Layers and Geometries for Extraction . . . 115
Layer . . . 115
Geometry Specification . . . 115
Create Pins . . . 116
Working with the Layer Assignment Tables . . . 116
Specifying the Extraction Limits . . . 122
Specifying the Depth . . . 122
Specifying the Distance . . . 122
Specifying the Width . . . 123
Ignoring Cells at the Top Level . . . 123
Excluding Metal Purpose Blockages in Layout Designs . . . 123
Signal Tab . . . 125
Relationship between Signal and Antenna Tabs . . . 126

© 2020 All Rights Reserved.

Extracting Signal Nets 128
Assigning Layers for Signal Extraction 129
Specifying the Extraction Limits 131
S pecifying the Creation of Must-Join Pins 132
Power Tab 135
Extracting Power Nets 136
Assigning Layers for Power Extraction 136
Specifying the Extraction Limits 138
Specifying the Creation of Must-Join Pins 138
Antenna Tab 139
Antenna Calculation by Abstract Generator 140
Updating the Technology File 142
Performing Process Antenna Calculations 142
Assigning Layers for Antenna Regions 151
Assigning Layers for Antenna Extraction 155
Comparing Geometry Specifications for Antenna Regions and Antenna Extraction 156
Excluding Signal Nets from Antenna Calculation 156
General Tab 157
Specifying the Existence of Connectivity Information in Design 158
Specifying the Connectivity Between Layers 158
Considerations for Specifying Layer Connectivity 159
Assigning Layers for Pin Geometry 159

5
Abstract Step 163
Multi-Patterning Technology (ICADVM20.1 Only) 164
Adjust Tab 166
Creating Boundary Pins 167
Creating Ring Pins 170
Creating Follow Ring Pins 171
Grouping Signal Nets 171
Grouping Power Geometries 172
Setting Options for CLASS CORE Ports 175
Creating CLASS BUMP Ports 181
Specifying Power and Ground Rail Characteristics 181

© 2020 All Rights Reserved.

Generating Abstracts for Routed Blocks with Power Rings 188
Blockage Tab 190
Assigning Layers for Blockages 191
Impact of Cutout and Spacing Options on the Blockage Types 198
Specifying the Size of the Window to be Cut Around a Pin 202
Specifying the Effective Width of Blockages 202
Supporting Pin Cutouts on Metal Layers Above the Pin Layer (for Stacked Vias) . . 202
Creating Routing Channels when Generating Cover Obstructions 203
Automatic Pin Stretching 204
Density Tab 207
Specifying the Metal Layers for Density Calculation 208
Specifying the Dimensions of the Density Window 209
Storage Location of Metal Density Information 211
Fracture Tab 212
Enabling Fracturing Options 212
Setting Up Options for 45-Degree Geometry 213
Site Tab 216
Specifying a Site Name 218
Defining a Site 218
Calculating Site Pattern 219
Overlap Tab 220
Creating an Overlap Layer in OpenAccess 220
Creating Overlap Boundary 221
Specifying Layers and Geometries for the Overlap Boundary 221
Fine Tuning the Overlap Boundary 221
Retaining Layout Information in the Abstract View 224
Creating Fill Blockages Automatically on Specific Layers 225
Bus Annotation in Abstract Generator 226
Supporting EXCEPTPGNET LEF Attribute 229
Supporting Layout Dual View 230
Advantages of Layout Dual View 230
Annotating Abstract Data in Layout Dual View 231
Abstract Generator options for Layout Dual View 232
Support for Copying Mixed Signal Routing Constraints from Layout to Abstract View . 232
Suppressing Messages Issued by Abstract Generator 234

© 2020 All Rights Reserved.

6
Integrated Abstract Generator 235
Launching the Integrated Abstract Generator 236
Customizing the Interface 236
Generating Abstracts 239
Loading Values in the Abstract Generator Form 239
Loading the Options File 240
Running the Abstract Generator Steps 240
Pins Step 240
Extract Step 241
Abstract Step 243
Viewing Results 244
Viewing the Cell Report 245
Viewing Log Information 246
Viewing the List of Cells 247
Environment Variables 249
optionsFile 250
showAbstractSection 251
showAntennaOptions 252
showExtractGspecTables 253
showExtractSection 254
showPinsSection 255
showOverlapBoundaryOption 256
showPinGroupingOptions 257
Form Descriptions 258

7
Dynamic Abstract Generation 261
Prerequisites for Dynamic Abstract Generation 262
Dynamic Abstraction Generation Use Model 263
Dynamic Abstract Generation Commands 264
Abstracting Pcells 265
Unabstracting Pcells 279
Highlighting Unabstracted Pcells 280

© 2020 All Rights Reserved.

Marking or Unmarking Pcells for Abstraction 281
Navigator Assistant (Using the Context-Sensitive Menu) 281
Layout Canvas (Using the Context-Sensitive Menu) 283

8
Multi-Patterning Technology Support in Abstract Generator (ICADVM20.1 Only) 285
This chapter discusses the following topics: 285
Overview 286
Prerequisites 288
Color Representation for Pins 289
Color Representation for Blockages 294
Other Supported Blockage Colors 296
Support for Trim Metal 297

A
Abstract Generator Menu Command Reference 301
Form Features 302
File Menu 303
Creating a Library 304
Opening a Library 306
Attaching Technology Information to a Library 308
Importing GDSII Information 309
Importing OASIS Data 312
Importing LEF 315
Importing DEF 318
Importing Logical Data 321
Importing Options File 323
Exporting LEF 324
Exporting a Report 327
Exporting Options File 328
Recording Commands and Events 329
Replaying Commands 331
Specifying Global Options 332

© 2020 All Rights Reserved.

Viewing the Statistics Report . . . 337
Closing Abstract Generator . . . 339
Bins Menu . . . 340
Adding a User Bin . . . 341
Renaming a User Bin . . . 342
Deleting a User Bin . . . 343
Setting Bin Options . . . 344
Cells Menu . . . 351
Selecting Cells Based on a Criteria . . . 352
Selecting All Cells . . . 354
Deselecting All Cells . . . 355
Selecting Cells with Invalid Views . . . 356
Sorting Cells . . . 357
Distributing Cells into Bins . . . 359
Moving Cells Across Bins . . . 362
Viewing and Editing Cell Properties . . . 363
Viewing and Editing Cell Terminal Properties . . . 366
Viewing Cell Status Report . . . 371
Editing Cellviews in Virtuoso Layout Editor . . . 372
Flow Menu . . . 374
Running the Pins Step . . . 375
Running the Extract Step . . . 376
Running the Abstract Step . . . 377
Running Form . . . 378
Help Menu . . . 379
Help Topics . . . 379
Menu Commands . . . 379
SKILL Commands . . . 379
Tutorial . . . 379
Error Messages . . . 379
About Abstract Generator . . . 379

B
Abstract Generator SKILL Functions . . . 381

List of Abstract Generator SKILL Functions . . . 381

© 2020 All Rights Reserved.

Abstract Generator SKILL Functions Overview . . . 385
Record File . . . 385
Entering Functions . . . 385
Completing Functions . . . 385
Importing . . . 387
absImportGDS . . . 388
absImportOasis . . . 390
absImportLogical . . . 393
absImportLEF . . . 395
absImportDEF . . . 397
absImportCTLF . . . 399
absImportVerilog . . . 400
absImportOptions . . . 401
Exporting . . . 402
absExportLEF . . . 403
absExportReport . . . 405
absExportOptions . . . 406
Library . . . 407
absGetLibrary . . . 408
absSetLibrary . . . 409
absAttachTechLib . . . 410
Selecting . . . 411
absSelect . . . 412
absSelectAllBins . . . 415
absSelectCells . . . 416
absSelectBin . . . 417
absSelectBinFrom . . . 418
absSelectCell . . . 419
absGetSelectedCells . . . 420
absSelectCellFrom . . . 421
absSelectCellsInList . . . 422
Deselecting . . . 423
absDeselectAllBins . . . 424
absDeselectCells . . . 425
absDeselectBin . . . 426
absDeselectBinFrom . . . 427

© 2020 All Rights Reserved.

absDeselectCell 428
absDeselectCellFrom 429
absDeselectCellsInList 430
Bin Operations 431
absGetBins 432
absGetBinType 433
absGetSelectedBins 435
absMoveSelectedCellsToBin 436
absRenameBin 437
absNewBin 438
absDeleteBin 439
absDeleteBinMoveCellsTo 440
absCopyBinOptions 441
Options 442
absGetOption 443
absSetOption 445
Bin Options 450
absGetBinOption 451
absSetBinOption 452
Flow Functions 453
absPins 454
absExtract 463
absAbstract 471
absVerify 493
Cell Properties 497
absGetCellProp 498
absSetCellProp 499
Terminal Properties 501
absGetTerminalProp 502
absSetTerminalProp 503
Miscellaneous 505
absDisableUpdate 506
absDistributeCells 507
absEnableUpdate 510
absExit 511
absRevalidateSelectedCells 513

© 2020 All Rights Reserved.

absSort 514
absVersion 515
iagCreateMenuItem 516
iagGenAbstract 517

C
Library Design Guidelines 519

General Guidelines 521
General Rules for All Cell Types 521
Designing Standard Cells 521
Designing Pins for Cells 526
Designing IO and Corner Cells 531
Designing Blocks 531
Determining Routing Grid Pitch 532
General Guidelines for Setting Pitch 532
Setting Pitch for Layers with the Same Preferred Direction 535
Setting Pitch for Via Stacking 536
Designing a Gridded Library for Innovus 537
Converting an Existing Library 538
Modeling Pins 539
Deciding Pin Layers 539
Power to Blockage Violations 539
Virtual Tracks 540
Centering On-grid Pins 541
Pins with Potential Routing Problems 542
Blockages 543
Blockages and Wiring Density 543
Blockages and Pin Accessibility 545
On-grid Versus Off-Grid Blockages 546
Cell Porosity and High Density 547
Routing Pitch 547
Pin Spacing 550
Pin Staggering 551
Defining Tracks for Wide Wire Designs 552
Variable-Height Versus Multiple-Height Cells 553

© 2020 All Rights Reserved.

Non-rectangular Blocks 556
Summary 557

D
Using Regular Expressions in Abstract Generator 559
Regular Expression Syntax as Used in Abstract Generator 559
Special Characters Supported by Abstract Generator 560
Special Constructs for “\” Supported by Abstract Generator 562
Reference 564

E
Abstract Generator Error and Warning Messages 565
ABS-10500 to ABS-10599: Library Messages 566
ABS-10000 to ABS-10099: Technology Messages 567
ABS-11000 to ABS-11099: Cell Messages 585
ABS-11800 to ABS-11899: Layout Messages 587
ABS-12000 to ABS-12099: Pin Messages 588
ABS-13000 to ABS-13099: Extract Messages 595
ABS-14000 to ABS-14099: Abstract Messages 601
ABS-16000 to ABS-16199: Import/Export Messages 610
ABS-17000 to ABS-17099: Messages 613
ABS-11900 to ABS-11999: System Messages 615
ABS-11500 to ABS-11599: OAToolbox Messages 620
ABS-20000 to ABS-20099 621
ABS-23000 to ABS-23099 622

F
Tutorial: Generate Abstracts by Using the Standalone Abstract Generator 623
Sample Library Data 623
License Requirement 623
Virtuoso Design Environment to LEF Flow Overview 623
Opening a Cell Library and Checking Technology Data 625
Starting Abstract Generator 625

© 2020 All Rights Reserved.

Opening the Library . . . 626
Generating Abstract Views . . . 627
Running the Pins Step . . . 627
Running the Extract Step . . . 627
Running the Abstract Step . . . 630
Exporting Abstract LEF . . . 632

G
Technology File Requirements for Abstract Generation . . . 635
Handling Technology Information in the Virtuoso Design Environment . . . 636
Technology Information in the Virtuoso Design Environment . . . 636
Creating a New Library . . . 636
Importing an Existing Library . . . 637
Modifying Technology Information . . . 637
Technology File Requirements . . . 638
Layer Definitions . . . 639
Via Definitions . . . 640
Physical Rules . . . 641
Application of the Minimum Area Rule for Pins . . . 643
Devices . . . 643
Voltage Dependent and Layer-Purpose-Based Rules . . . 644
Specific Requirements for Antenna Calculation . . . 645
Specific Requirements for Metal Density Calculation . . . 649
Specific Technology Requirements for Viewing Data In Layout . . . 649
Specific Requirements for Specifying Place-and-Route Information . . . 650
Potential Issues Related to the Technology File in the Virtuoso Design Environment . . . 652
Working with Read-Only Technology File . . . 653
Setting DATABASE MICRONS in the Exported LEF . . . 653
Including USEMINSPACING Properties in the Exported LEF . . . 654
Fewer than Two Metal Layers Defined . . . 655
Layer Not Fully Defined . . . 655
Layer with Spacing Rules is a Derived Layer . . . 655
Via Layers Not Fully Defined . . . 656
Upper Layer is Not a Routing Layer . . . 657
Antenna Values Must Not Equal Zero . . . 657

© 2020 All Rights Reserved.

Minimum Width Value is Zero 657
Minimum Width is Not a Multiple of the Manufacturing Grid 657
Minimum Spacing Value Equals Zero 658
Cut Layer Must Have Upper and Lower Routing Layer 658
Specifying Routing Grids 659
Converting Common Lisp Files to SKILL or Tcl Format 659

Index 661

© 2020 All Rights Reserved.

Preface

Virtuoso® Abstract Generator is a library modeling tool that lets you create abstracts for standard cells, macro blocks, and IOs from detailed layout information in Library Exchange Format (LEF), Design Exchange Format (DEF), Graphics Design Station II (GDSII) Stream, and OASIS formats.

This user guide is aimed at developers and designers of integrated circuits and assumes that you are familiar with:

- The Virtuoso design environment and application infrastructure mechanisms designed to support consistent operations between all Cadence® tools.
- The applications used to design and develop integrated circuits in the Virtuoso design environment, notably, the Virtuoso Layout Suite, and Virtuoso Schematic Editor.
- The Virtuoso design environment technology file.
- Component description format (CDF), which lets you create and describe your own components for use with Layout XL.

This preface contains the following topics:

- Scope
- Licensing Requirements
- Related Documentation
- Additional Learning Resources
- Customer Support
- Feedback about Documentation
- Typographic and Syntax Conventions

© 2020 All Rights Reserved.

Scope

Unless otherwise noted, the functionality described in this guide can be used in both mature node (for example, IC6.1.8) and advanced node and methodologies (for example, ICADVM20.1) releases.

Label	Meaning
(ICADVM20.1 Only)	Features supported only in ICADVM20.1 advanced nodes and advanced methodologies releases.
(IC6.1.8 Only)	Features supported only in mature node releases.

© 2020 All Rights Reserved.

Licensing Requirements

Standalone Abstract Generator

The following figures describe the licensing requirements for standalone Abstract Generator:

© 2020
All Rights Reserved.

Important

Colored designs are supported only in the ICADVM20.1 release, with the 95011 license.

© 2020
All Rights Reserved.

Integrated Abstract Generator

The following image captures the licensing behavior of the integrated Abstract Generator.

IC6.1.8 release: Required licenses

- DFII framework '111' license
- One of the following licenses:
  - Virtuoso Layout Suite XL license + 2 GXL tokens
  - Virtuoso Layout Suite GXL (4 tokens) + 2 GXL tokens
  - Virtuoso Layout Suite EAD

ICADVM20.1 release: Required licenses

- DFII framework '111' license
- Virtuoso Advanced Node Framework '95011' license
- One of the following:
  - Virtuoso Layout Suite XL license + 2 GXL tokens
  - Virtuoso Layout Suite GXL (4 tokens) + 2 GXL tokens
  - Virtuoso Layout Suite EXL

© 2020 All Rights Reserved.

> **Note:** Additionally, when a library is opened, if the technology file has data reflecting geometries of 20nm, 16nm, or 14nm, the `Virtuoso_Adv_Node_Opt_Lay_Std` license (95512) is checked out. If the technology file has data reflecting geometries of 10nm or smaller, the `Virtuoso_Adv_Node_Opt_Layout` license (95511) is checked out.

Abstract Generator does not check out the Virtuoso Layout Suite EAD/EXL license like other licenses. It only checks out the EAD cockpit.

For information about licensing in the Virtuoso design environment, see *Virtuoso Software Licensing and Configuration Guide*.

Related Documentation

What's New and KPNS

- *Virtuoso Abstract Generator What's New*
- *Virtuoso Abstract Generator Known Problems and Solutions*

Installation, Environment, and Infrastructure

- *Cadence Installation Guide*
- *Virtuoso Design Environment User Guide*
- *Virtuoso Design Environment SKILL Reference*
- *Cadence Application Infrastructure User Guide*

Technology Information

- *Virtuoso Technology Data User Guide*
- *Virtuoso Technology Data ASCII Files Reference*
- *Virtuoso Technology Data SKILL Reference*
- *Virtuoso Technology Data Constraint Reference*

© 2020 All Rights Reserved.

Virtuoso Tools

- *Virtuoso Layout Suite SKILL Reference*
- *Virtuoso Layout Suite L User Guide*
- *Virtuoso Parameterized Cell Reference*
- *Component Description Format User Guide*
- *Design Data Translator's Reference*
- *Virtuoso Multi-Patterning Technology User Guide*

Relative Object Design and Inherited Connections

- *Virtuoso Relative Object Design User Guide*
- *Virtuoso Schematic Editor L User Guide*

Additional Learning Resources

Video Library

The Video Library on the Cadence Online Support website provides a comprehensive list of videos on various Cadence products.

To view a list of videos related to a specific product, you can use the *Filter Results* feature available in the pane on the left. For example, click the *Virtuoso Layout Suite* product link to view a list of videos available for the product.

You can also save your product preferences in the Product Selection form, which opens when you click the *Edit* icon located next to *My Products*.

Virtuoso Videos Book

You can access certain videos directly from Cadence Help. To learn more about this feature and to access the list of available videos, see Virtuoso Videos.

© 2020 All Rights Reserved.

Rapid Adoption Kits

Cadence provides a number of Rapid Adoption Kits that demonstrate how to use Virtuoso applications in your design flows. These kits contain design databases and instructions on how to run the design flow.

In addition, Cadence offers the following training course on Virtuoso Abstract Generator:

- Abstract Generator

To explore the full range of training courses provided by Cadence in your region, visit Cadence Training or write to training_enroll@cadence.com.

Note: The links in this section open in a separate web browser window when clicked in Cadence Help.

Help and Support Facilities

Virtuoso offers several built-in features to let you access help and support directly from the software.

- The Virtuoso *Help* menu provides consistent help system access across Virtuoso tools and applications. The standard Virtuoso *Help* menu lets you access the most useful help and support resources from the Cadence support and corporate websites directly from the CIW or any Virtuoso application.
- The Virtuoso Welcome Page is a self-help launch pad offering access to a host of useful knowledge resources, including quick links to content available within the Virtuoso installation as well as to other popular online content.

  The Welcome Page is displayed by default when you open Cadence Help in standalone mode from a Virtuoso installation. You can also access it at any time by selecting *Help – Virtuoso Documentation Library* from any application window, or by clicking the *Home* button on the Cadence Help toolbar (provided you have not set a custom home page).

For more information, see Getting Help in *Virtuoso Design Environment User Guide.*

Customer Support

For assistance with Cadence products:

- Contact Cadence Customer Support

© 2020 All Rights Reserved.

Cadence is committed to keeping your design teams productive by providing answers to technical questions and to any queries about the latest software updates and training needs. For more information, visit https://www.cadence.com/support.

- Log on to Cadence Online Support

  Customers with a maintenance contract with Cadence can obtain the latest information about various tools at https://support.cadence.com.

Feedback about Documentation

You can contact Cadence Customer Support to open a service request if you:

- Find erroneous information in a product manual
- Cannot find in a product manual the information you are looking for
- Face an issue while accessing documentation by using Cadence Help

You can also submit feedback by using the following methods:

- In the Cadence Help window, click the *Feedback* button and follow instructions.
- On the Cadence Online Support Product Manuals page, select the required product and submit your feedback by using the *Provide Feedback* box.

© 2020 All Rights Reserved.

Typographic and Syntax Conventions

The following typographic and syntax conventions are used in this manual.

text	Indicates names of manuals, menu commands, buttons, and fields.
`text`	Indicates text that you must type as presented. Typically used to denote command, function, routine, or argument names that must be typed literally.
`z_argument`	Indicates text that you must replace with an appropriate argument value. The prefix (in this example, *`z_`*) indicates the data type the argument can accept and must not be typed.
`|`	Separates a choice of options.
`{ }`	Encloses a list of choices, separated by vertical bars, from which you **must** choose one.
`[ ]`	Encloses an optional argument or a list of choices separated by vertical bars, from which you **may** choose one.
`[ ?argName` *`t_arg`* `]`	Denotes a *key argument*. The question mark and argument name must be typed as they appear in the syntax and must be followed by the required value for that argument.
`...`	Indicates that you can repeat the previous argument.
	Used with brackets to indicate that you can specify zero or more arguments.
	Used without brackets to indicate that you must specify at least one argument.
`,...`	Indicates that multiple arguments must be separated by commas.
`=>`	Indicates the values returned by a Cadence® SKILL® language function.
`/`	Separates the values that can be returned by a Cadence SKILL language function.

If a command-line or SKILL expression is too long to fit within the paragraph margins of this document, the remainder of the expression is moved to the next line and indented. In code excerpts, a backslash (\ ) indicates that the current line continues on to the next line.

© 2020 All Rights Reserved.

1

Overview of Abstract Generator

Abstract Generator is a library modeling tool that lets you create abstracts for standard cells, macro blocks, and IOs from detailed layout information in Library Exchange Format (LEF), Design Exchange Format (DEF), and Graphics Design Station II (GDSII) Stream formats.

Important

LEF/DEF version 5.8: References to LEF and DEF version 5.8 functionality and features apply only for Abstract Generator in the Innovus™ Digital Implementation System (Innovus) releases. LEF/DEF 5.8 is not available with Innovus.

An abstract is a high-level representation of a layout view. An abstract contains information about the type and size of cells, position of pins or terminals, and the overall size of blockages. The abstracts generated are based on physical layout and logical data, process technology information, and specific cell-modeling requirements. The abstracts are used in place of full layouts to improve the performance of place-and-route tools, such as Innovus. After the place-and-route is complete, the abstracts are replaced back with the layouts.

The functionality for abstract generation is available through **two interfaces**, which access the same executable. You can choose to access a particular interface based on your requirements, the current design environment, and the familiarity with an interface. You can invoke the executable either as a standalone application or from within the Virtuoso design environment.

- The standalone Abstract Generator interface can be used to generate abstracts for a single library at a time. You can run this application both in GUI and non-GUI mode. The graphical interface of this application is modeled on the abstract generation flow steps — Pins, Extract, and Abstract. This interface provides advanced options and is designed for experienced users.
- The integrated Abstract Generator interface launched from the Virtuoso design environment can be used to generate abstracts for individual or multiple cells of a library. This interface is simple and is designed for novice users who are either not familiar with or find the advanced options of the standalone application difficult to comprehend. Being tightly integrated with the Virtuoso design environment, this interface allows users

© 2020 All Rights Reserved.

working on layouts in the Virtuoso design environment to continue with abstract generation for their layouts in the same environment.

This user guide describes Abstract Generator, a library modeling tool for creating abstracts from detailed layout information for standard cells, macro blocks, and IOs. This guide also describes the integrated Abstract Generator interface used for creating abstracts in the Virtuoso design environment for specific cells.

This chapter covers the following topics:

- Use Model for Abstract Generation
- Block Override Support in Abstract Generation
- Flows Supported by Abstract Generator

© 2020 All Rights Reserved.

Use Model for Abstract Generation

Abstract Generator generates LEF abstracts for cells in cell libraries. A cell library typically consists of standard cells but can also include IO and corner cells and some small custom blocks. The individual components of the library are described in an appropriate layout format, such as GDSII or LEF.

Abstract Generator lets you import layout information and create abstracts for some or all of the cells in the library. You can then reference any of the library cells in your design, safe in the knowledge that a suitable abstract exists for that cell. After generating abstracts for library cells, you should not have to return to the tool unless the library is revised either in response to some change in the fabrication process or through the addition of new cells or modification of the existing ones.

Abstract Generator satisfies the requirements of library designers and production engineers by presenting an uncluttered flow through the library development process.

The more experienced user enjoys maximum flexibility in entering cell information, partitioning cells into exclusive sets, fine tuning option settings, and iterating through the abstract generation process. The tool also provides reliable support for moving backward in the flow and clearing feedback on potential and real problems with the abstracts generated.

The less experienced designer is provided with a simple flow through the essential steps required to update a library with predefined settings. The more flexible and user-friendly interface, that is, the integrated Abstract Generator, which can be launched in the Virtuoso design environment is also available at the disposal of the novice users.

Block Override Support in Abstract Generation

Abstract Generator supports overridden terminals and instance terminals in block domain. The Abstract Generation commands do not create overriding terminals or instance terminals. However, these overridden terminals and instance terminals can be present in OpenAccess data coming from Innovus or any other application using OA. Therefore, the support is required for interoperability between Innovus and Virtuoso.

The behavior of commands in Abstract Generator is given as follows:

1. The commands processing pins and pin figures on overridden terminals and instance terminals, of master instance having pins and pin figures, should consider overridden terminal and instance terminal as a normal block domain terminal or instance terminal, without any difference in the behavior. For example, the Pin Optimize command should optimize pins on overridden terminal and instance terminal.

© 2020 All Rights Reserved.

2. The commands involving change of signal type should ensure that nets connected to overriding terminals and instance terminals can only be of signal type `power`, `ground`, `tieHi`, `tieLo`, and `tieOff`.

   In Abstract Generator, you can set the signal type for terminals only as `signal`, `clock`, `power`, `ground`, or `analog`. If pins on the layout and the terminals for the same pins have another signal type, such as `reset`, the signal type value is retained as `reset` in the abstract view as well.

3. The commands processing (deleting) empty nets should be careful because a net may be empty only in the block domain and may have a terminal/instance terminal in module domain. Therefore, the net is not empty as corresponding terminals and instance terminals are hidden in the block domain.

Commands related to terminals and instance terminals are as follows:

- Abstract Generate

Concept of Overriding Terminals and Instance Terminals

The "*tieHigh*", "*tieLow*", "*tieOff*", "*power*", "*ground*", and "*signal*" nets specified in verilog can be implemented in block domain using any physical-only net. The example for this is given as follows:

```
/ Verilog file for cell "top_level" view "layout"
// Language Version: 2001
module top_level (
    A,
    B,
    C,
    D,
    Z
    );
    input A;
    input B;
    input C;
    input D;
    input _IN;
    output Z;
    output _OUT;
    wire net1;
    assign _OUT = 1'b1;
    assign _IN = 1'b0;
```


© 2020 All Rights Reserved.

```
    block3 I1 ( .A(A), .B(B), .C(C), .Y(net1), .Z(1’b1));
    block I2 ( .A(net1), .B(1’b1), .C(D), .Z(Z), .„nd! (1’b0), .\vdd! (1’b1));
endmodule // top_level
```

In the mentioned example, observe the following in the connectivity:

- The terminals _IN and _OUT are connected to tie0 and tie1nets, respectively.
- The instance terminals corresponding to I1/Z and I2/B are connected to tie.
- The instance terminals corresponding to I2/vdd! and I2/gnd! are connected to tie1 and tie0 nets, respectively

It is significant to understand that verilog designer has no clue how 1’b1 (tieHi) and 1’b0 (tieLow) nets will be implemented in the layout or block domain. It depends on the block designer to choose any physical only net to implement 1’b1 and 1’b0. For example, the following array of nets can be used to implement.

- _IN and I2/gnd! (tie0) to be implemented by net VSS (physical only).
- _OUT and I2/vdd! (tie1) to be implemented by net VDD (physical only).
- I1/Z and I2/B (tie1) to be implemented by net INTERNALVDD (physical only).

The following changes in the block domain will have to be made.

- Hide _IN _OUT terminals.
- Hide instance terminals I2/gnd!, I2/vdd!, I1/Z and I2/B.
- Create overriding terminals and overriding instance terminals with same names to implement them using the above mentioned physical only nets.

© 2020 All Rights Reserved.

Flows Supported by Abstract Generator

The following flows describe the interaction between the standalone Abstract Generator interface, and the tools and translators on the Virtuoso platform:

- GDSII to LEF
- DEF to OA
- LEF to OA
- Virtuoso Layout Flow

GDSII to LEF

In this flow, Abstract Generator imports GDSII data and generates the abstract LEF 5.x. This LEF abstract is then used to create layouts, which can be imported directly into the Virtuoso environment.

The abstract generation step is usually performed by a library or an IP developer and is independent of any floorplanning.

DEF to OA

In this flow, Abstract Generator imports DEF layout and LEF abstract views and generates an abstract layout.

Note: The imported LEF is required to create the abstract views of cells, which are then instantiated when the DEF is read by Abstract Generator.

© 2020 All Rights Reserved.

This flow lets you convert a routed block generated by Innovus into a cellview suitable for use in the Virtuoso design environment. You can also generate a LEF version of the block abstract if required.

LEF to OA

In this flow, LEF layout information is imported to generate an abstract layout view.

This flow lets you import an existing LEF abstract generated by a non-Cadence tool and use the advanced features of Abstract Generator to render the data for use in Cadence floorplanning and place-and-route engines, particularly power routers. For example, you might use Abstract Generator to form correct power geometry groupings into `LEF PORTS` for `sroute`. For more information about power geometry groups, see Grouping Power Geometries.

© 2020 All Rights Reserved.

The abstract view can be used directly in the Virtuoso design environment or you can export LEF 5.x for use with Innovus.

Virtuoso Layout Flow

This flow facilitates a tight integration between Abstract Generator and Virtuoso Floorplanner, which bypasses the import and export LEF steps used in the LEF to OA flow.

In this flow, all layout data creation and abstract generation stay within the Virtuoso design environment. There is no requirement for LEF. This part of the flow can be done by using Virtuoso Floorplanner or Abstract Generator. The DEF side of the flow requires a netlist and can only be done by using Virtuoso Floorplanner.

© 2020 All Rights Reserved.

Note: Virtuoso Floorplanner can invoke Abstract Generator to generate an abstract view for either a layout or a softLayout view.

© 2020
All Rights Reserved.

© 2020 All Rights Reserved.

2

Standalone Abstract Generator

This chapter describes the standalone Abstract Generator interface that is designed to be sufficiently intuitive to allow first-time users to make good progress and at the same time provide the flexibility required to satisfy the goals of expert users. The interface provides comprehensive status information and supports the recommended use model by allowing backward steps in the flow and providing helpful feedback at all stages.

This chapter describes how to launch Abstract Generator and introduces the flow used by the tool to generate abstracts from layout data. It touches upon the commands and options and some of the graphical elements to describe the features you will encounter while using the interface. Finally, the chapter provides guidelines to optimize the performance of Abstract Generator.

Note: Some Abstract Generator forms might not fit your screen. For best results, set your screen resolution to 1280 x 1028 or above.

This chapter includes the following sections:

- Getting Started
- Launching the Standalone Abstract Generator
- Getting Familiar with the Interface
- Customizing the Interface
- Generating Abstracts
- Running Abstract Generator in Batch Mode
- Log, Replay, and Record File Behavior
- Saving and Loading the Options and Status Files
- Optimizing the Performance of Abstract Generator

© 2020 All Rights Reserved.

Getting Started

Setting Up the Abstract Generator Environment

Abstract Generator reads a file named `.abstractrc` during the startup process.

Note: If you use the command line options to launch Abstract Generator, the tool first loads the `.abstractrc` file and then the replay file that has been specified using the `-replay` argument.

When the tool starts, it searches for the `.abstractrc` file at the following locations in the order specified:

- *`cds_install_dir`*`/tools/dfII/local`
- Current directory (.)
- Home directory (~)

You can modify the `.abstractrc` file to customize the tool for a group of users. You can also specify your custom standard cell SKILL code in the `.abstractrc` file so that Abstract Generator reads it every time you launch the tool. In addition, the `.abstractrc` file can enable you to run the required Tcl or SKILL replay file when you launch the tool.

Note: Do not set up library options in the `.abstractrc` file because they are overwritten when you open a library.

© 2020 All Rights Reserved.

Launching the Standalone Abstract Generator

You can launch the standalone Abstract Generator directly from the installation directory.

To launch the standalone interface, do the following:

1. Ensure that you have the following directory in your path:

   `your_install_dir/tools/dfII/bin`

2. Move into the run directory. You can run Abstract Generator from any directory, but preferably not from the release hierarchy.

3. Type `abstract` at the command line prompt. You can specify various arguments with the `abstract` command, such as:

```
abstract
    [-replay filename]
    [-log filename]
    [-tcl]
    [-nogui]
    [-V | -W]
    [-dbType]
    [-cdslib path]
    [-help]
```

   These arguments are described below.

`-replay filename`	Specifies a Tcl or SKILL replay file to be run automatically when Abstract Generator launches. The format of the replay file depends on the extension language setting. By default, Abstract Generator operates in the SKILL mode. For more information, see Inspect the Results.
`-log filename`	Specifies the name of the log file to be generated for the current session. By default, the log file is named `abstract.log`. For more information, see Inspect the Results.
`-tcl`	Sets the extension language to Tcl. This means that Abstract Generator recognizes only Tcl commands in its command line and replay file. By default, Abstract Generator operates in the SKILL mode.
`-nogui`	Disables the graphical user interface. Abstract Generator operates directly from the command line prompt in the shell window.

© 2020 All Rights Reserved.

`-V | -W`

Displays version information.

```
% abstract -V
Abstract version 5.2.2 03/23/05 15:10 (hostname)
```

Displays subversion information.

```
% abstract -W
sub-version ICOA5.2.51.14
```

Tip

When reporting problems, provide the information returned by the `-W` argument to customer support.

`-dbType`

Displays the name of the database used.

```
% abstract -dbType
OPENACCESS
```

`-cdslib` *`path`*

Specifies the relative or complete path to a `cds.lib` file. Abstract Generator reads the `cds.lib` to read the list of libraries to be loaded.

`-help`

Displays the list of arguments for the `abstract` command.

© 2020 All Rights Reserved.

When you type the `abstract` command at the command line prompt, the Abstract Generator form is displayed.

© 2020
All Rights Reserved.

Getting Familiar with the Interface

The standalone Abstract Generator interface includes the following components to help you accomplish a wide range of tasks.

- Main Window
- Menu Bar
- Toolbar
- Bin Pane
- Cell Pane
- Log Window
- Command Entry Field
- Command History Window

When you run any abstract generation flow step, Abstract Generator lets you monitor the progress of the step through a message box. (See Progress Window)

Abstract Generator lets you copy and paste text between GUI fields and tables. To copy text from a *GUI field,* click-and-drag to select the required text. To paste the text in its new location, click the middle-mouse button in the target field or table cell.

To copy text from a *table cell*, click in the table cell containing the text to be copied. The text for the entire cell is attached to the clipboard until another GUI element is selected. To paste the text in its new location, click the middle-mouse button in the target field or table cell.

Partial text selection for copy and paste is available only in non-table GUI fields.

The chapter also discusses options for doing the following using the standalone GUI:

- Customizing Fonts
- Customizing the Appearance
- Customizing the Abstract Generator Flow Steps

© 2020 All Rights Reserved.

Main Window

When you start Abstract Generator for the first time, the main window appears.

Main Window

The application title bar contains the name of the application and the name of the current library if one has been set. If no library has been opened, the title bar of the application displays *no current library* next to the name of the application, as is shown in the figure above.

© 2020
All Rights Reserved.

Menu Bar

The menu bar provides access to all Abstract Generator operations. You can also perform some of these operations by quickly using the buttons on the toolbar below the menu bar.

Menu commands and toolbar buttons are grayed out when they are not available or do not apply in a particular scenario. For example, until you set a library you cannot import logical or layout information, so these and most of the other commands will be grayed out. Similarly, you cannot use the abstract generation steps until you have selected one or more cells in the *Cell Pane*.

The menu bar provides the following menus:

- File Menu
- Bins Menu
- Cells Menu
- Flow Menu
- Help Menu

© 2020 All Rights Reserved.

File Menu

The commands in the *File* menu let you specify all setup information, ranging from defining technology information and specifying a library through importing physical and logical data for the cells and blocks to be processed, for Abstract Generator to process a library. The commands for specifying this information are also available on the toolbar.

Other commands in the *File* menu let you export LEF descriptions of abstracts, generate a statistics report for the current library, and save the option settings of the current session. You can also replay a previous session.

For detailed information about the commands in this menu, see File Menu.

Bins Menu

Use the commands in the *Bins* menu to create, rename, and delete user bins, and to access the *Bin Options* form.

For more information about these menu commands, see Bins Menu.

Cells Menu

Use the commands in the *Cells* menu to select and sort the cells displayed in the cell table, move cells between different bins, view cell status reports, and set cell and terminal properties. The *Edit* submenu opens the Layout Editor, which you can use to inspect or modify any cellview.

For more information about these menu commands, see Cells Menu.

Flow Menu

The *Flow* menu provides access to the main abstract generation flow commands: Pins, Extract, and Abstract. You can also access these commands by using the last four buttons on the toolbar.

Note: The flow options are available only when technology data has been successfully validated by the Abstract Generator technology checker and you have selected one or more library cells to be processed.

For more information about the commands in this menu, see Flow Menu.

© 2020 All Rights Reserved.

Help Menu

The commands in the *Help* menu let you access help system of Abstract Generator.

From this menu, you can select the *Error Messages* submenu if you require more descriptive error or warning message information. This information can also be accessed directly from the Log Window by clicking any available hyperlink.

For more information about the commands in this menu, see Help Menu.

Toolbar

The toolbar is located below the pull-down menus and provides seven button shortcuts to the most commonly used functions in Abstract Generator.

© 2020
All Rights Reserved.

Moving from left-to-right, the first three buttons are associated with functions in the *File* menu and the last three with functions in the *Flow* menu.

The left-to-right sequence of the buttons implies the basic flow in the abstract generation process, that is, set the current library, import data, and then run the abstract generation steps. You can use these functions in any permitted sequence to achieve more complex flows. Illegal button sequences are prevented by graying-out unavailable functions.

Refer to the following links to know more about the functions of these buttons:

- Opening a Library
- Importing GDSII Information
- Importing Logical Data
- Running the Pins Step
- Running the Extract Step
- Running the Abstract Step

© 2020 All Rights Reserved.

Bin Pane

The *Bin* pane exists on the left of the main application window below the menu bar and toolbar.

The *Bin* pane displays the system bins and user bins, if any, and the number of cells in each bin. The number next to each bin name in the *Bin* pane indicates the number of cells in that bin. In the figure above, all the bins are empty.

Bins

Bins are like buckets for placing cells. Each cell in a library is always contained in one bin. Each bin is associated with certain options, which allows the cells in a library to be distributed into mutually exclusive sets for processing. For example, Abstract Generator uses the option settings for `binA` to determine how to treat the cell `cellA` in `binA`.

There are essentially two types of bins, systems bins and user bins.

© 2020 All Rights Reserved.

System Bins

There are five *system bins,* one for each of the four main types of cells — *Core, IO, Corner, Block* — and one to store any cells that you do not want to process at a particular time — *Ignore*. Any cells placed in the *Ignore* bin are not deleted but are excluded from the set of cells to be processed.

Note: You cannot add, rename, or remove system bins.

User Bins

If you require bins of any given type in addition to the systems bins, you can create *user bins*. For example, you might want to create additional bins so that you can experiment with different option settings on two or more sample sets of cells. Or you might want to divide the core cells in your library into two separate sets to process some cells using slightly differing option settings and compare the results generated. You can do this up by creating a user bin and specifying its type as *Core*, and then applying option settings to it different from those of the *Core* bin.

User bins differ from system bins only in that you can delete and rename user bins, while system bins are permanent and cannot be renamed or deleted.

Creating User Bins

You can create user bins in addition to the existing system bins. To create a user bin, load a library and choose *Bins – Add* from the menu bar. Enter a name and set a type for the new bin by selecting a choice from the *Choose bin type* drop-down box. The bin type is important because it determines the default values for certain options. When you click *OK*, the new bin is added to the bottom of the list in the *Bin* pane.

User bins differ from system bins only in that you can delete and rename user bins, while system bins are permanent and cannot be renamed or deleted.

Note: The commands in the *Bins* menu apply only to user bins.

Working with Bins

When you select a bin, you see the contents of that bin listed in the Cell Pane.

© 2020 All Rights Reserved.

If you select two or more bins simultaneously, the contents of all the selected bins are displayed in the *Cell* pane. This gives you complete flexibility in defining the set of cells to be processed without compromising on the different options set for the cells.

When you select multiple bins, you can use the *Bin* column in the *Cell* pane to recognize the bin to which a cell belongs.

© 2020
All Rights Reserved.

Cell Pane

The *Cell* pane is situated on the right of the main window below the menu bar and toolbar. It lists the cells contained in the selected bins and reports the status of those cells.

You need to select at least one cell before you can run any of the flow steps or view a status report on a cell.

Note: For more sophisticated selection operations using criteria like the cell name, height, properties, and view status, use the *Cells – Select* command.

This section describes the following aspects of working with the *Cell* pane:

- Types of Cellviews
- Viewing Status Information
- Generating Cell Reports

© 2020 All Rights Reserved.

Types of Cellviews

When a cell library containing physical or logical information is imported into Abstract Generator, the view information about the cells is represented in the appropriate column of the *Cell* pane. Similarly, when a cell is processed through the various Abstract Generator steps – Pins, Extract, and Abstract – cellviews are created on successful completion of each step.

Abstract Generator uses the following 5.x library format cellviews:

- `layout` for layout views
- `logical` for logical views
- `abstract.pin` for Pins views
- `abstract.ext` for extracted views
- `abstract` for abstract views

Viewing Status Information

The *Cell* pane is divided into a number of columns, one for each of the views handled by Abstract Generator: *Layout, Logical, Pins, Extract,* and *Abstract*. A symbol in any of these columns indicates the presence of that view for the selected cell.

Note: An additional column, *Bin*, appears as the first column when more than one bin is selected in the *Bin* pane. The *Bin* column displays the name of the bin to which the cell belongs.

© 2020
All Rights Reserved.

The symbol displayed indicates the status of that view, as summarized in the following table.

Symbol	What It Represents
✓	A green tick indicates that the view is valid.
!	An amber exclamation mark indicates that the view was generated with warnings.
✗	A red cross indicates that the view was generated with errors.
?	A black question mark indicates that the view is present but was not created by Abstract Generator.

There is also a grayed-out version of each symbol, which indicates that the cellview in question has been rendered invalid, typically because of a change to one of the option settings or because of the regeneration of a predecessor view when an earlier step in the flow is re-run.

For example, a gray cross against a cell means that the view in question was originally generated with errors but has now been rendered invalid.

Note: To maximize the usefulness of the feedback, cell status is not propagated from one view to another. For example, the pins view for a particular cell can show warnings against it, while the extract view for the same cell has none.

© 2020 All Rights Reserved.

Generating Cell Reports

A cell report provides detailed information about the status of a cell with explanations about the messages generated for different views. To view a cell report, select the cell or cells for which you wish to view the report and choose *Cells – Report* from the menu.

If you select more than one cell, the arrow buttons at the bottom of the form allow you to move backward and forward through the selected set. In the above figure, the arrow buttons are not enabled because only one cell has been selected in the *Cell* pane.

Log Window

The *Log* window is located in the bottom half of the Abstract Generator main window. The log displays all user interactions with Abstract Generator, including the display of version information on startup, error, warning, and info messages, and Tcl/SKILL commands that have been input during a session.

© 2020 All Rights Reserved.

Above the *Log* window, Abstract Generator provides the interface language *Interpreter* Options, *Tcl* or *Skill*. *Skill* is the default and recommended language option.

The *LOG*, *ERROR*, *WARNING*, and *INFO* entries in the *Log* window are color coded to make it easier to locate serious problems. The *LOG* and *INFO* entries are highlighted in BLUE, *WARNING* messages in ORANGE, and *ERROR* messages in RED.

Each *ERROR*, *WARNING*, and *INFO* message also has a unique message identifier, such as (ABS-246). In this identifier, “ABS” represents “Abstract Generator”, and “246” represents the unique number for identifying the message. All *ERROR* and *WARNING* message identifiers are highlighted in clickable BLUE hyperlinks. Clicking the identifier invokes the Cadence Help (cdnsHelp) in your browser window that displays extended description and possible workaround for the particular *ERROR* or *WARNING* message you clicked on.

For more information about error messages, see Abstract Generator Error and Warning Messages.

Command Entry Field

The TCL/SKILL command entry field is at the bottom of the *Log* window and extends up to under the *Command History* window. In this field, at the `abstract>` prompt, you can

© 2020 All Rights Reserved.

manually enter a specific command function, for example `absSelectAllBins`, that tells Abstract Generator what operation to perform. The *Command History* window provides other quicker methods of entering command information in this field.

For more information about all the SKILL commands that can be entered in this field, refer to Abstract Generator SKILL Functions.

Command History Window

The *Command History* window provides a list of all the Tcl/SKILL commands that have been entered in the command entry field.

This window helps you save time in re-typing a command. Double-click to select a command in this window, then place your cursor over the Command Entry Field, and click the middle mouse button to copy the command in the command entry field.

Alternatively, you can also scroll through the commands listed in the *Command History* window by using the cursor keys on the keyboard, with whatever command you have highlighted appearing in the command entry field ready for use.

© 2020 All Rights Reserved.

Progress Window

The *Percent Complete* message box lets you monitor the progress of the current step.

The label on the *Percent Complete* message box tells you what operation is currently being tracked, for example *Running Extract* as in the figure below, while the bar and percentage give you a visual indication of the progress.

Use the *Stop* button in the *Percent Complete* message box to stop the current operation. Abstract Generator prompts you to confirm that you stop the operation.

Depending on the step that is running, Abstract Generator might not be able to stop the operation immediately. In this case, it will proceed until the next safe point before stopping. You can ascertain the current state of your library from the status indicators in the *Cell* pane, and from the log file saved in the run directory.

© 2020 All Rights Reserved.

Customizing the Interface

You can customize the fonts and appearance of the standalone Abstract Generator interface by editing the `.Xdefaults` file. This section also explains how to apply the customized changes. Finally, this section demonstrates how to customize the Abstract Generator flow steps.

Customizing Fonts

The following defaults are used by Abstract Generator:

```
*Abstract*font: Helvetica -12
*Abstract*Text*font: Fixed -14
*Abstract*Entry*font: Fixed -13
*AbstractBubbleHelp*font: Helvetica -12
```

The sequence is significant. `*Abstract*font` is the most general rule, so it must be specified before the rules for text and entry widgets so that they can override the general value. If `*Abstract*font` is placed below either one of these, then it will override the values specified by the text and entry rules. The sequence has no effect on the bubble help rule because it uses a separate class, `AbstractBubbleHelp`, to store its resources.

To change all the fonts used in Abstract Generator, add the following line to your `.Xdefaults` file:

```
*Abstract*font: <font specifier>
```

Caution

Take care when specifying fonts. If a font is incorrectly specified, Abstract Generator will not initialize correctly.

The two recommended ways of specifying fonts are described below.

1. **Standard X Logical Font Descriptor (XLFD)**

   This is used by all X applications and takes the following format:

   ```
   -foundry-family-weight-slant-setwidth-addstyle-pixel-point-resx-resy-spacing-
   width-charset-encoding
   ```

   For example,

   ```
   -*-Lucidatypewriter-Medium-R-*-*-*-120-*-*-*-*-iso8859-1
   ```

   The individual options are each separated by a hyphen (`-`). An asterisk (`*`) indicates no value for that field.

© 2020 All Rights Reserved.

Tip

You can obtain this string using the `xfontsel` utility in a shell window. To use `xfontsel`, use the drop-down menus to select the font required, then click the *select* button to copy the specification to the clipbord.

2. **Tcl font descriptors**

   The Tcl font descriptor requires only the font family, size, and any modifiers (bold, italic, and so on).

   For example,

   ```
   <font family> <size> [<descriptor>+]
   ```

   Note: The sequence is significant; the size must come before the descriptor list.

   Possible descriptor values are

   ```
   normal bold roman italic underline overstrike
   ```

   These values have standard meanings and might not apply for all fonts. For example, fixed-width fonts might not often have an italic variants.

   Examples,

   - `Helvetica -12`
   - `{Times New Roman} -14 bold italic`

     Use braces to enclose a font family that features a space. The hyphen prefixing the font size is required.

To specify different fonts for different GUI widgets, add the following lines to your `.Xdefaults` file:

- **Entry widgets**

  For single-line entry boxes, typically used for entering numbers in the *Bin Options* form, use

  ```
  *Abstract*Entry*font: <font specifier>
  ```

- **Text widgets**

  For multiple-line text input boxes, used for almost all other textual input in Abstract Generator, use

  ```
  *Abstract*Text*font: <font specifier>
  ```

- **Bubble Help**

  For bubble help text, use

© 2020 All Rights Reserved.

```
*AbstractBubbleHelp*font: <font specifier>
```

Customizing the Appearance

You can specify the color scheme for Abstract Generator UI items and controls, such as tables and buttons, in the `.Xdefaults` file.

The following defaults are used by Abstract Generator:

```
*Abstract*foreground: black
*Abstract*background: gray52
*Abstract*Button.background: gray52
*Abstract*Table.foreground: black
*Abstract*Table.background: medium slate blue
*Abstract*Text.foreground: black
*Abstract*Entry.foreground: black
*Abstract*Label.foreground: black
*Abstract*Label.background: gray52
```

This section describes options to specify the color scheme for the following:

- Buttons
- Tables
- Text Entries
- Entry Boxes
- Labels
- GUI

© 2020 All Rights Reserved.

Buttons: Use the following options for customizing the colors of buttons and tabs:

- This foreground option applies the color setting to the foreground (text) of all buttons and tabs.

  ```
  *Abstract*Button.foreground: <color>
  ```

- This background option applies the color setting to the background of all buttons and tabs.

  ```
  *Abstract*Button.background: <color>
  ```

In the figure below, the text (foreground) on the two active buttons and the four tabs is set to `red` and the button and tab background is set to `blue`.

Tables: Use the following options for setting the color scheme for tables:

- This option applies the color setting to the foreground (text) of the selected row(s).

  ```
  *Abstract*Table.foreground: <color>
  ```

- This option applies the color setting to the background of the selected row(s).

  ```
  *Abstract*Table.background: <color>
  ```

Important

To view the applied color effect, you must select a row.

© 2020 All Rights Reserved.

The figure below illustrates the `black` background and `red` foreground table setting. The applied color effect is showing only for the selected rows.

Another example of the same table color setting is provided in the figure below, which captures the *Open Library* form.

The *Geometry Specification* column in the *Layer Assignment* tables is also considered to comprise of rows of a table. The text at the current position of the cursor assumes the applied

© 2020 All Rights Reserved.

table foreground color. An example is shown in the figure below in which the table foreground color is set to `red` and the cursor is in the last row in the *Geometry Specification* column.

Text Entries: This option applies the color setting to non-numeric text entries and fields expecting non-numeric user text inputs.

```
*Abstract*Text.foreground: <color>
```

In the figure below, the text color has been set to `blue`.

© 2020
All Rights Reserved.

Entry Boxes: This option applies the color setting to drop-down lists (such as the drop-down lists in the *Layer* column in the *Layer Assignment* tables) and combo boxes, fields with numeric values, and fields expecting numeric inputs.

```
*Abstract*Entry.foreground: <color>
```

The following figure shows the effect of setting the entry box color to `magenta`. The setting for text entry (`blue`) has been retained from the previous example to illustrate the comparison between text entries and entry boxes.

The figure below shows the effect of the same entry box color setting (`magenta`) on drop-down lists.

© 2020
All Rights Reserved.

Labels: Use the following options for setting the color of the foreground and background of labels:

- This option applies the color setting to the foreground of all labels. It overrides the `*Abstract*foreground` setting wherever applicable.

  ```
  *Abstract*Label.foreground: <color>
  ```

- This option applies the color setting to the background of all labels.

  ```
  *Abstract*Label.background: <color>
  ```

In the figure below, the label foreground is set to `magenta` and background to `yellow`.

© 2020
All Rights Reserved.

GUI: Use the following options for setting the color of the foreground and background of the GUI:

- This foreground option applies the color setting mainly to the GUI text and all active options.

  `*Abstract*foreground: <color>`

- This foreground option applies the color setting to the GUI background.

  `*Abstract*background: <color>`

In the figure below, the GUI foreground is set to `red` and the background to `black`.

© 2020 All Rights Reserved.

Applying the Font and Appearance Changes

After you have added the font or the color scheme specification to your `.Xdefaults` file, merge the changes into the X resource database (XRDB) using one of the following commands:

- This command will overwrite the previously applied changes.

  ```
  xrdb < <path of the .Xdefaults file>/.Xdefaults
  ```

- This command will apply over the previously applied change, even if the setting has been removed from the `.Xdefaults` file.

  ```
  xrdb -merge <path of the .Xdefaults file>/.Xdefaults
  ```

Queries to the XRDB return only the values present when the `.Xdefaults` file was last read in. However, the `.Xdefaults` file is usually read in at login on most UNIX systems.

© 2020 All Rights Reserved.

Customizing the Abstract Generator Flow Steps

Tasks that you otherwise cannot perform directly by using the available Abstract Generator options can be executed using an advanced methodology. This methodology is based on the concept of calling custom code, also called a hook, during the abstract generation flow. Abstract Generator uses some predefined hooks, which can be customized, to extend the functionality of the existing Abstract Generator flow steps.

This section briefly describes the predefined hooks and how to customize them.

A hook is a predefined function or procedure that you can redefine. Abstract generator provides predefined hooks, which are automatically called before and after the execution of a flow step (Pin, Extract, or Abstract). There are therefore, two predefined hooks for every flow step. Following are the predefined hooks in Abstract Generator:

- `PinsPreHook`: This function is called prior to the execution of the Pins step.
- `PinsPostHook`: This function is called after the execution of the Pins step.
- `ExtractPreHook`: This function is called prior to the execution of the Extract step.
- `ExtractPostHook`: This function is called after the execution of the Extract step.
- `AbstractPreHook`: This function is called prior to the execution of the Abstract step.
- `AbstractPostHook`: This function is called after the execution of the Abstract step.

Before the `PinsPreHook` function is called, the pin and blockage objects in the abstract are converted into corresponding pin and blockage shapes.

Note: After execution of the `AbstractPostHook` function, these shapes are converted into the corresponding objects.

© 2020 All Rights Reserved.

The following diagram depicts the sequence of execution of the predefined hooks:

These hooks are empty by default and you can redefine them to execute a specific task every time a flow step runs.

These predefined hooks exist both in SKILL and Tcl. Both SKILL and Tcl hooks, empty or redefined, are called for each flow step irrespective of the interpreter selected in the GUI. This ensures consistent results regardless of the interpreter chosen for abstract generation.

Note: For each step, SKILL hooks are called prior to the Tcl hooks.

© 2020 All Rights Reserved.

You can modify the definition of the predefined hooks. Ensure to retain the name of the predefined hook while redefining it.

Note: The return value of a hook is not used.

The hooks should take four arguments in the following format:

```
t_binName t_libName t_cellName t_viewName
```

Abstract Generator passes appropriate values for these arguments at run time.

© 2020 All Rights Reserved.

Example of a Customized AbstractPostHook Function

Consider a scenario in which you need to copy the original pin shape from a layout to the pin purpose in the abstract view. This task cannot be directly performed using any of the available Abstract Generator options. You can, however, use the `AbstractPostHook` function to define this functionality so that it executes automatically after each run of the Abstract step.

Type the following SKILL `AbstractPostHook` code at the `abstract` prompt.

```
procedure(AbstractPostHook(binName libName cellName abstractViewName)
;; This function restores the original pin shapes from a layout in the
;; final abstract generated by Abstract Generator
let((cvL cvA fig layoutViewName)
   layoutViewName = absGet("ViewLayout")
   ;; Opening the layout view
   cvL = dbOpenCellViewByType(libName cellName layoutViewName)
   ;; Opening the abstract view
   cvA = dbOpenCellViewByType(libName cellName abstractViewName "" "a")
   when( cvL && cvA
        ;; Corresponding to each terminal in the layout view, looking for
        ;; terminals in the abstract view
        foreach( term cvL~>terminals
            termA = dbFindTermByName(cvA term~>name)
            ;; Copying pin figures and creating pin shapes corresponding
            ;; to each terminal located in the abstract view
            foreach( pin term~>pins
                fig = dbCopyFig(pin~>fig cvA)
                dbCreatePin(termA~>net fig)
            )
        )
    )
))
```

By default, `AbstractPostHook` option runs before the pin shapes and blockage shapes present in an abstract are translated to the corresponding pin and blockage objects. Consider that there are some shapes on the Metal1 layer in the layout view. When creating a detailed blockage to cover these Metal1 layer shapes, before the post hook runs, these blockage shapes will be created on (Metal1 boundary). To refer to these shapes, you have to refer to the shapes on (Metal1 boundary) LPP. After running post-hook, Abstract Generator converts the Metal1 blockage shapes to Metal1 blockage objects. Thereafter, the blockages can be accessed using the corresponding object ID.

© 2020 All Rights Reserved.

Use the `runAbstractHookPostOat` option to specify whether `AbstractPostHook` needs to be called before or after conversion of pin and blockage shapes into corresponding pin and blockage objects. By default, the `runAbstractHookPostOat` option is set to false. So, `AbstractPostHook` is called before the conversion.

When set to true, `AbstractPostHook` is called after the conversion.

```
absSetOption("runAbstractHookPostOat" "false")
```

For details about various database functions (prefixed with `db`), see Chapter 2 in the *Virtuoso Design Environment SKILL Reference*.

© 2020 All Rights Reserved.

Generating Abstracts

This topic outlines the basic steps required to start generating abstracts for cells or libraries. Click any of the steps to see their elaborate descriptions.

Load a Library

You can either open an existing design library (see Opening a Library) or create a new one (see Creating a Library).

To load an existing library, use the *Open Library* form, accessible from the *File – Library – Open* menu. This form lists the libraries that are read in from the `cds.lib` file located in the run directory on startup. Select the library you want to process. The library path tells Abstract Generator where to find existing information about the library and where to store the intermediate and final views created during abstract generation.

© 2020 All Rights Reserved.

Attach Technology Information

Next, you can attach a technology library (see Attaching Technology Information to a Library) to the design library you have selected to process.

If you attach a technology library to the design library or if there is a technology library associated with the design library, Abstract Generator uses the existing technology library information, such as the `.abstract.options` file (see Saving and Loading the Options and Status Files).

Import Data

The selected design library may already contain either layout or logical views of cells or views created by Abstract Generator in a previous session. If it does not, you will need to import physical layout or logical data. Therefore, after specifying the cell library and attaching the necessary technology data, import information about the cells for which you want to create abstracts. You can import various types of data, such as LEF, DEF, GDSII, and OASIS.

The figure below shows the types of layout and logical data that can be imported into Abstract Generator and the format of data that is exported.

© 2020
All Rights Reserved.

You can import physical layout information, typically represented in the GDSII, OASIS, LEF, or DEF formats. Physical layout information comprises hierarchical shape databases that describe the polygons, paths, and text that make up each individual cell. This data is translated and a layout view for each cell is created in the OpenAccess database.

You can also import logical information, which is represented in Verilog or Liberty format (LIB) formats. This type of information includes data on which pins should be created and the terminal directions assigned. By default, Verilog has input and output terminals defined, whereas LIB has more number of terminals defined including tristate, clock, power, and ground.

Tip

You can use the *Terminal Properties* form to change these settings.

Abstract Generator creates logical views in the current library directory. If there is already a layout view present for any given cell, Abstract Generator does not create a new cell but adds the new logical view to any other views already present for the cell. If logical information is imported after running the Pins Step, Abstract Generator confirms that the layout and logical views match before extracting terminal properties from the logical view.

For more information about importing data, see the import options in the File Menu.

Distribute Cells

Abstract Generator uses the concept of Bins to distribute the library cells into a number of mutually exclusive sets.

When you import cell information into Abstract Generator, it is by default assigned to the *Core* bin. This might be sufficient for certain libraries, but there will be cases where you require a more sophisticated distribution before you start processing your cells.

You can use the *Select* and *Sort* functions of Abstract Generator to locate the cells you are interested in, and then use the manual and automatic *Distribute* functions to *Move* these cells into the appropriate system or user bins. These commands are available on the Cells Menu.

Create Abstracts

Whether you are generating abstracts for an entire library or for a few hierarchical blocks, the approach is typically the same. You can initially focus on a small subset of cells or blocks, establish option settings for this subset, and then process the remaining cells or blocks in a single run.

© 2020 All Rights Reserved.

1. Identify the cells with which you want to experiment, and begin to Generate Abstracts for these cells.
2. Inspect the Results in the generated abstracts.
3. You can then begin to Modify Option Settings as required, and then rerun the required steps.
4. If you are satisfied with the results, you can process the remaining cells in your library.

Generate Abstracts

Each of the three main flow steps–Pins, Extract, and Abstract–have their own set of options that control the way in which any cell is processed. You can make your initial option settings either before you start generating abstracts or when you run any of the individual steps.

Inspect the Results

After you have generated abstracts for your sample of selected cells, you must evaluate whether the results are satisfactory. Abstract Generator provides various features to help you verify your abstracts. You can inspect the results using the following interfaces:

- **Cell Pane:** The first source of result evaluation comes from the *Cell Pane* in the main window. Abstract Generator uses color-coded symbols in the *Cell* pane to indicate the result of a particular abstract generation flow step. A color-coded symbol against a cell corresponding to a particular step indicates if the step completed without problems or whether warnings or errors were generated during the step. For more details about color-coded symbols, see Viewing Status Information.

  The *Cell* pane area also indicates when a particular view becomes invalid, for example through a change in an option setting.

Cell	Layout	Logical	Pins	Extract	Abstract
BUF2	✓				
BUF4	✓		✓		
DFFRHQS1	✓				
XOR2S1	✓				
cds_logo	✓		×		
counter3	✓		!		
divider	✓		✓	✓	×
phaseDetPath	✓		!	✓	
phaseDetector	✓				

- **Layout:** If you want to see a detailed graphical representation of any view, you can use VLS XL. You can use this software to examine the pin and blockage geometry generated, the sizings and spacings applied, and to make minor edits.

© 2020 All Rights Reserved.

To launch the *Virtuoso Layout Suite XL Editing* window, select *Cells – Edit* and select a view.

- **Report File and Log File:** You can use the information files generated by Abstract Generator to examine potential problems with abstracts.

  All the messages issued by Abstract Generator are stored in a log file (`abstract.log`) in the run directory. If multiple users are working in the same cell directory, the output is written to the file `abstract.log.1` up to the file `abstract.log.9`. For more information, see Inspect the Results.

  For more structured information, you can export a report file categorizing the cells in the library according to various criteria. You can also view online information about any cell in your library.

Modify Option Settings

After you assess the abstracts, modify the option settings as required and rerun the relevant steps to generate a new abstract.

Note: When you change options related to a particular step, the view created by that step is invalidated along with all “downstream” views present for that cell.

Tip

You can revalidate invalid views by using the *Cells – Revalidate Selected* menu command.

When you are satisfied with the results of your tests and the options you want to use for each bin, you can process the remaining library cells either bin-by-bin, flow step-by-flow step, or all at a time.

Export Abstracts

When you finish generating abstract cellviews, you can export this data in the form of LEF abstracts for use in place-and-route tools.

The menu command *File – Export – LEF* provides the functionality to translate cell abstracts into LEF, which can be used as input to place-and-route tools. For more information about exporting abstract LEF, see Exporting LEF.

The menu command *File – Export – Options* lets you save bin options settings for reuse in a later session. This saves all the current option settings to a `.abstract.options` file in

© 2020 All Rights Reserved.

your current working directory. For more information about export options, see Exporting Options File.

Note: Abstract Generator writes the following system files in the library directory:

- `.abstract.options`: Stores Abstract Generator option settings for the library. This file is available only in the SKILL format.
- `.abstract.cellstatus`: Stores the Abstract Generator status for all cells in the library. For example, a cellview might have the status `exist`, `exists with errors`, `exists with warnings`, and so on.

Running Abstract Generator in Batch Mode

During an abstract session, every command and menu option you use is recorded in the `abstract.record` file. You can use this file as the replay file for the next session in either GUI or non-GUI mode. The replay file can be used to rerun the previous session or run a new session in batch mode. For this, you only need to change the library and cell names in the replay file.

Note: To save your previous work and to avoid the `abstract.record` file from being overwritten, the `abstract.record` file is renamed as `abstract.replay`, and then it is replayed. A new `abstract.record` file is created for the current session.

To run Abstract Generator in batch mode:

- For GUI mode, run the following command at the command line prompt.

```
% abstract -replay filename
```

- For non-GUI mode, run the following command at the command line prompt.

```
% abstract -nogui -replay filename
```

Abstract Generator runs the steps in sequence as per the replay file settings. After the batch run is complete, a library is created in the Library Manager, and the logical, layout, and abstract views of the selected cells are imported into the library. An LEF file is also generated depending on the selection of the GUI or non-GUI mode.

Note:

- It is recommended to remove any `.abstract` files between the GUI and non-GUI replay runs.
- In the non-GUI mode, it may be necessary to explicitly set the default bin in the replay file.

```
absSetOption("DefaultBin" "Core")
```


© 2020 All Rights Reserved.

This command must be inserted before the `absSelectCell()` call so that the selected cells can be imported into the default bin.

- In the `abstract.record` (replay) file, you can define the same cell and terminal properties for multiple cells belonging to a single bin or different bins. For this, you only need to add a `foreach` statement with the selected cells. For example,

```
absExtract()
celllist_for_prop_add = cellList("cell1" "cell2")
cellList=absGetSelectedCells()
cellList=parseString(cadr(parseString(cellList ":")))
         foreach(cell cellList
                   absSetCellProp(cell "symmetry" "X Y R90")
                   absSetCellProp(cell "prCellClass" "areaIO")
                   absSetTerminalProp( cell "VDD" "shape" "abutment")
                   absSetTerminalProp( cell "VSS" "shape" "abutment")
                  );foreach
absAbstract()
absExportLEF()
```

Here, you must run the Pin and Extract steps before selecting the cells for defining the cell and terminal properties.

- While using the replay file, Abstract Generator generates LEF abstracts for cells of both the previous and current sessions. If you want LEF abstracts for only the current session, do one of the following:
  - Move the cells of the previous session to the *Ignore* bin.
  - Move the cells of the current session to a different bin, generate LEF abstracts only for the cells of that bin, and then place the cells back in the original bin.
- In an abstract session, the bin-specific and step-specific options are saved in the `.abstract.options` file of a library. If you do not want these settings in batch mode, update the `abstract.record` (replay) file so that the library defaults are skipped. The `.abstract.options` file does not exist if you have not run Abstract Generator on the library.

Video

For a quick overview of this feature, see Running Virtuoso Abstract Generator Using Replay File.

© 2020 All Rights Reserved.

Related Topics

Log, Replay, and Record File Behavior

Recording Commands and Events

Replaying Commands

Log, Replay, and Record File Behavior

During an abstract session, the following types of files are created:

- **abstract.log file:** Stores all the information commands and the info, warning, and error messages issued by Abstract Generator during the current session. Instead of using the default abstract.log file, you can use the -log option to specify a different log file name.

  Example: To create a log file named session.log, specify the following:

  ```
  % abstract -log session.log
  ```

  If there are multiple active abstract sessions in the same testcase directory, then separate log files with names such as abstract.log, abstract.log.1, and abstract.log.2 are created.

- **abstract.log.cdslck file:** Along with a log file, an abstract.log.cdslck file is also created for each new abstract session. An abstract.log.cdslck file indicates that an abstract session is currently running, for which the log file is being used. The abstract.log.cdslck file exists only for the current abstract session, and is deleted when you exit the session. The abstract.log is overwritten if there is no abstract.log.cdslck file present in the current directory. If you specify a different log name for a session, then the corresponding <user_defined>.log.cdslck is created.

  Example: For session1.log file, a corresponding session1.log.cdslck file is created. If a session1.log.cdslck already exists, and you invokes an abstract session by specifying:

  ```
  % abstract -log session1.log
  ```

  In such cases, an error is displayed and Abstract Generator is not invoked. To invoke Abstract Generator, you either need to specify a different log file name or close the currently active session and try again.

- **abstract.record:** Every command and option you use during the current abstract session is recorded in a .record file. The numbering of the abstract.record and abstract.log files is always in sync.

© 2020 All Rights Reserved.

- **abstract.replay:** For every abstract session, an abstract.record file is created. You can use the abstract.record file from a previous run as the replay file for the current session by specifying:

  ```
  % abstract -replay abstract.record
  ```

  To save your previous work and to avoid the abstract.record file from being overwritten, the abstract.record file is renamed as abstract.replay, and then it is replayed. A new abstract.record file is created for the current session.

In all cases, the abstract.log, abstract.record, and abstract.replay files will have the same numbering to identify the correct abstract shell.

Specifying the CDS.log File Name

To specify a unique log file name using the CDS_LOG_VERSION environment variable, set the variable before starting the Cadence software or include the `setenv` command in your `.cshrc` file:

```
setenv CDS_LOG_VERSION {sequential | pid}
```

Here:

- `sequential` adds a sequential number to the name of the log file, such as CDS.log.1 or CDS.log.2
- `pid` adds the number of the UNIX process to the name of the log file, such as CDS.log.1719 or CDS.log.2250

Filtering Messages

You can use the following option to filter redundant messages. Only info and warning messages are suppressed with this option.

```
absSetOption(suppressMessage <id1> <id2> <id3> …)
```

Abstract Generator suppresses the messages with the message IDs mentioned in `suppressMessage`. The suppressed messages are not displayed in the Abstract Log window and not stored in the `abstract.log` file. An info message is displayed for the first time for every suppressed message. At the end of the abstract step, a table displays the messages that have been suppressed along with their suppression times during the current session.

© 2020 All Rights Reserved.

Note: Error messages are not suppressed with the above option. If an error message ID is mentioned in `suppressMessage`, then a warning message displays stating that this is an error message and will not be suppressed.

© 2020 All Rights Reserved.

Saving and Loading the Options and Status Files

Abstract Generator saves the options file (`.abstract.options`) in the `.abstract/<library name>` directory in your current working directory. It no longer saves the file in the current design library. The `.abstract` directory is created in your current working directory when Abstract Generator runs.

Consider, for example, that you are working with the design library `myLib` and your current working directory is `/home/user1`. In this case, Abstract Generator saves the options file in `/home/user1/.abstract/myLib/.abstract.options`. Abstract Generator also saves the existing options file as `.abstract.options.prev` in the same directory where the `.abstract.options` file is saved.

Abstract Generator loads the options file in a specific sequence.

- It first loads the options file (`.abstract.options`) from the `<current working directory>/.abstract/<library name>` directory.
- If not available in the above location, it reads the file first from the attached technology library directory and then from the design library directory.

If the same options are available in the next file loaded, the options are overwritten accordingly.

Note: A message highlighted in red is displayed in the *Log* window, informing you about the new default location of the options file.

Abstract Generator saves the status file (`.abstract.status`) in the `.abstract/<library name>/cellStatus` directory in your current working directory. It no longer saves the file in the cell directory. Information about each cell is stored in a separate status file. The status file stores the following information:

- Name of the active cell in the selected bin
- View statuses (Pins, Abstract, or Extract View) of all cells in the selected bin

© 2020 All Rights Reserved.

- Status of the cell when it is moved from one bin to another

The status file is read when a library is loaded. The file is updated when the pins, extract, and abstract steps are run, and also when the library is closed.

```
cUiSelectCellP {test1}
cUiSetViewStatus {test1} {layout} {1}
cUiSetViewStatus {test1} {logical} {0}
cUiSetViewStatus {test1} {abstract.pin} {3}
cUiSetViewStatus {test1} {abstract.ext} {1}
cUiSetViewStatus {test1} {abstract} {3}
```

Note: For backward compatibility, Abstract Generator copies information from the `.abstract.status` file (at the cell level) to the `.abstract.status.cellName` file (at the user directory level). Any further changes are done in the `.abstract.status.cellName` file.

© 2020 All Rights Reserved.

Optimizing the Performance of Abstract Generator

Some abstract generation functions can adversely affect the performance of the tool and should be run only when necessary. You can keep the following guidelines in mind while using the standalone Abstract Generator interface:

- Do not extract signal and power nets for blocks or IO cells unless you are sure you need to do so. For example, you must extract signal nets if you want to perform antenna extraction, while you might need to extract power nets if you want to create ring pins.
- If you do need to extract signal or power nets,
  - Do not create pins on a given layer unless it is necessary. For example, do not create pins on polysilicon, diffusion, or contact layers, except where there are dominant cut-layer spacings.
  - Define the `Poly` layer as `Poly and Oxide` and `Oxide` layer as `Oxide andnot Poly` in the geometry specifications under the *Extract signal nets* and *Extract power nets* sections in the Extract step.
  - You may not require the *Maximum depth* value to be 32 as per your design scenario under the *Extract Limitations* section in the Extract step. Limit the extraction by the hierarchy depth.
  - Ensure in the Abstract step that not all of the extracted geometry is turned into pins. If you do not do this, Abstract Generator must cut around each pin, which increases the time taken to generate the blockage model. You can avoid this for power pins by creating ring or boundary pins and for signal pins by creating boundary pins.
- Turn off the *Fracture pins* and *Fracture blockages* options if you observe that Abstract Generator is taking a lot of time in the Abstract step.
- Do not create detailed blockages on contact or via layers for large blocks. The large number of contact obstructions slows down the Abstract step.
- Do not run grid analysis for blocks or IO cells.
- If blocks or IO cells have pins or obstructions with 45-degree shapes, you must either ensure that they are removed from the abstract (for example, if they are not in pins, then the cover blockage will hide them) or consider increasing the stair-step width.
- If pins and extract cellviews are not required by the user, the `QuickAbstract` option can be enabled to improve the performance of Abstract Generator by 30 percent on small- and mid-size designs. When the `QuickAbstract` option is set to true, then Abstract Generator will not generate the intermediate cellviews that is Pin and Extract cellviews. Only the Abstract cellview will be generated. By default, the `QuickAbstract` option is set to false.

© 2020 All Rights Reserved.

- Take care when creating geometry specifications to act on pins on the `prBoundary` layer. This layer can feature thousands of shapes and an incorrect geometry specification can cause these to be merged into a single rectangle for each pin.
- Ensure to correctly define the *Gate/Drain* region under the *Antenna* tab in the Abstract step for antenna calculations. If the *Use different layer assignments for antenna calculations only* option is turned on, define the `Poly` layer as `Poly` and `Oxide` layer as `Oxide andnot Poly` in the geometry specifications.
- Limit the *Geometry search depth* to a lower value if required, under the *Text* tab in the Pins step.
- If Abstract Generator is running out of memory while extracting nets on large designs and you get the following message:

  ```
  *WARNING* dbDeleteShape: A memory allocation for 1073741824 bytes
  failed. The process heap is 511MB, and OpenAccess is using 1405MB.
  The available system memory is 991MB, and physical system memory
  is 7857MB.
  ```

  Set the following option in the Abstract Generator console or script files.

  ```
  absSetOption("OptimizeBigDesignMemory" "true")
  ```

  This option takes care of memory issues that arise on the big block designs in Abstract Generator. By default, this option is set to `false`. To optimize the memory for big designs, set this option to `true`. Also, ensure that the abstract executable is run in 64-bit mode.
- If extraction is aborted because of an excessive number of shapes while extracting the power nets on large designs, increase the limit of shapes to be extracted using the following option. The default value is `10000`.

  ```
  absSetOption( "ExtractShapeLimit" "150000")
  ```


© 2020 All Rights Reserved.

The following two forms can be used for option settings:

Bin Options Form

You can access this form by using the *Bins - Options* menu command. In this form, you can view and modify all options associated with the entire abstract generation flow.

Running Form

Whenever you run any flow step, Abstract Generator opens the *Running step...* form. This form allows you to modify only the options that are relevant to the steps about to be run.

When you are satisfied with the options settings, use the four flow steps to generate abstracts for the selected cells. You can run the steps either one at a time or all at once for any or all of the cells.

Pins	In the Pins step, Abstract Generator creates a place-and-route boundary for the cell and the starting pin shapes for each of the nets to be extracted. It then matches the pins created against those described in any logical view present and appends the appropriate pin direction.
Extract	In the Extract step, Abstract Generator derives which shapes are connected to which nets by tracing the connectivity from the *pin* purpose shapes created during the Pins step. The tool creates a shape with purpose *net* in the top level of the extract view, and for each such shape creates a pin on the appropriate net. The overlap boundary is also calculated if required. Finally, Abstract Generator uses the antenna options to create library process antenna information for custom blocks and standard cells. The resulting antenna model helps mitigate the problem of gate damage caused by chemical and mechanical polishing.

© 2020 All Rights Reserved.

Abstract	In the Abstract step, Abstract Generator adjusts the pin shapes created during the Extract step to create the final shapes required by place-and-route tools. It then fractures these pin shapes into rectangles. Next, Abstract Generator applies a layer blockage model selected by the user to create the final blockage geometry in the abstract. The blockage geometry is then ally fractured into rectangles. It then removes from the abstract all layers other than those with purpose *pin*, *blockage*, or *boundary* and deletes the instance hierarchy. At this stage, all the required geometry is at the top level of the abstract.

Chapter 3, Pins Step, Chapter 4, Extract Step, and Chapter 5, Abstract Step describe these flow steps in detail.

© 2020 All Rights Reserved.

3

Pins Step

This chapter describes the first step in the abstract generation process: the Pins step. Usually, no pins are explicitly defined in the layout view, so they have to be generated before you can continue the abstract generation process. Abstract Generator derives pins from the text labels in the layout view and places the locations at the text origins.

Note:

- Where there is a logical view present, Abstract Generator only creates pins that match logical view terminals.
- Abstract Generator copies the placement status of the pins in the layout view to the abstract view. The supported placement states are `locked`, `placed`, `fixed`, `none`, and `unplaced`.
- Abstract Generator automatically detects shapes placed in width spacing pattern (WSP) regions and ignores these shapes while creating pins under labels and PR boundaries.

Abstract Generator provides a number of options that support a variety of layout and text mapping conventions. Options for mapping and for controlling the generation of pins are present on the Map Tab.

Abstract Generator performs the following functions during the Pins step:

- Maps text labels to terminal names and creates terminals
- Creates physical pin shapes corresponding to the geometry overlapping the text labels
- Creates the place-and-route boundary

To access the *Running step Pins* form, select one or more cells in the main window and click the *Pins* button. Alternatively, select *Flow – Pins*. This opens the *Running step Pins* form. You can select the options you need on this form and click the *Run* button. The *Percent Complete* message box reports the progress of the step.

When the Pins step is complete, Abstract Generator places a symbol against the cell in the *Cell* pane to indicate the status of the Pins view that was created. For information about these icons, see Viewing Status Information.

© 2020 All Rights Reserved.

This chapter describes the following tabs on the *Running step Pins* form:

- Map Tab
- Text Tab
- Boundary Tab
- Blocks Tab

In the *Running step Pins* form, use the radio buttons on the left to navigate to the tab pages for the different steps and bins you are using.

For information about other abstract generation steps, see the following:

- Extract Step
- Abstract Step

Preparing for Abstract Generation

Before beginning with the abstract generation process, you might need to perform the following operations:

- Open a library. (See Opening a Library)
- Attach technology information. (See Attaching Technology Information to a Library)
- Import layout information. (See Importing GDSII Information)
- Import logical information. (See Importing Logical Data)

Note: Not all of these procedures are compulsory; see individual procedure for details.

© 2020 All Rights Reserved.

Map Tab

To access the *Map* tab, select *Flow – Pins*. The *Map* tab is the first tab in the *Running step Pins* form.

© 2020
All Rights Reserved.

On the *Map* tab, you can specify the text labels to be used for creating starting pin shapes for each of the nets to be extracted. You can also specify the pins that are power, ground, clock, analog, or output pins.

The following options on the *Map* tab let you control how Abstract Generator creates starting pin shapes for each of the nets to be extracted:

- *Map text labels to pins*
- *Power pin names*
- *Ground pin names*
- *Clock pin names*
- *Analog pin names*
- *Output pin names*

Mapping Text Labels to Pins

In the *Map text labels to pins* field on the *Map* tab, you can specify the text labels in the layout view that are to be mapped to the pins on nets in the pins view. Specify the layer-purpose pairs based on which you want Abstract Generator to search for geometry of any given layer-purpose pair with a text label. The *Map text labels to pins* field is case-sensitive.

The syntax is as follows:

```
(textLPP1 geomLPP1 geomLPP2)
(textLPP2 geomLPP3 geomLPP4)
```

where,

- The syntax of `textLPP` is either `(textLayer textPurpose)` or `textLayer`.
- The syntax of `geomLPP` is either `(geomLayer geomPurpose)` or `geomLayer`.

Note: If you specify only a layer name instead of a layer-purpose pair, Abstract Generator considers all the purposes on that layer excluding the ones specified in the Ignored Purpose(s) field on the *General* tab (*File – General Options*).

© 2020 All Rights Reserved.

For each statement, Abstract Generator first identifies all the text labels on the specified `textLPP`. Then, for each occurrence of a text label that overlaps a shape on the first `geomLPP`, it creates a pin of the same size as the shape on the metal layer overlapping the text label. This is depicted in the figure below. The text label `A` on `metal1 pin` overlaps the shape `S1` on `metal1 drawing`. So Abstract Generator creates the vertical `PinA`, which is of the same size as `S1`.

In case the text label overlaps more than one shape, Abstract Generator creates a single merged pin for the overlapped shapes. As shown in the figure below, the text label `A` overlaps two intersecting shapes `S1` and `S2`. In this case, Abstract Generator merges the vertical and horizontal pin shapes and creates the single L-shaped `PinA`, which is cornered at left.

© 2020
All Rights Reserved.

Furthermore, Abstract Generator creates pins for all the text labels with the same name, irrespective of whether a shape has already been made into a pin by a previously evaluated text label. It also merges the overlapping pin shapes generated from the text labels with the same name. This is illustrated in the figure below where another text label with the same name `A` overlaps the shapes `S2` and `s3`. Even though Abstract Generator already created a pin on `S2` from the first text label `A`, it creates another pin shape from the overlap of the second text label `A` with the shapes `S2` and `S3`. The overlapping pin shapes in this situation result in a single merged U-shaped `PinA`.

Abstract Generator proceeds in the similar manner for all other text labels on `textLPP`. The order in which Abstract Generator analyzes shapes is from the first specified `geomLPP` to the next.

Consider the following examples as the text that can be specified in the *Map text labels to pins* field:

- `(textLayer (metal1 drawing) (metal2 drawing))`

  This means that text labels on `textLayer` for all purposes except the ignored purpose(s) are mapped first to shapes on layer `metal1` purpose `drawing` and then to shapes on layer-purpose pair `metal2 drawing`.

- `(textLayer metal1) (textLayer metal2)`

  This means that labels on `textLayer` for all purposes except the ignored purpose(s) are mapped first to shapes on `metal1` excluding the ignored purpose(s). Then, in a second pass, Abstract Generator maps labels on `textLayer` to shapes on `metal2` excluding the ignored purpose(s).

© 2020 All Rights Reserved.

> **Note:** To do this, Abstract Generator takes the first mapping, `(textLayer metal1)`, and creates appropriate pins on the `metal1` layer. If there are text labels overlapping `metal2` shapes, they are not matched by this rule and Abstract Generator issues a number of ABS-12006 warnings informing you about labels that could not be mapped to pins. These labels are picked up in the second pass.

Unless there are terminals already present, you must specify a value in the *Map text labels to pins* field to generate a valid pins view. If you leave the field empty, Abstract Generator does not generate pins in the abstract view and the warning message ABS-12004 is displayed. To generate these pins, specify valid values in the *Map text labels to pins field*, and rerun the Pins step.

Running the Pins Step for Routed Blocks

You do not need to specify a value in the *Map text labels to pins* field when running the Pins step on routed blocks.

Routed blocks already have terminals, so Abstract Generator does not create any new terminals. If the existing terminals have physical pins, as is usually the case with signal nets, these are retained in the Pins step.

Note: If these pins are not on valid LEF layers, Abstract Generator deletes these pins after generating the ABS-12023 message.

Otherwise, all the routed shapes attached to the net are turned into pins on the net. This is the case with power nets, which usually have terminals but no physical pins.

Writing Regular Expressions

All fields in which you specify power, ground, output, analog, and clock pin names accept regular expressions. The pin name in question is formed from the subset of the label text that matches the regular expression you enter.

Consider the following examples:

- To create power pins *only* from labels that match the string `VDD`, specify the following power pin name regular expression:

  ```
  ^VDD$
  ```

  The `^` matches the beginning of a string; the `$` matches the end of a string. Therefore, only strings containing the `VDD` match the regular expression.

© 2020 All Rights Reserved.

- To create a pin name that is identical to a label text containing the string `VDD` anywhere, specify `.*VDD.*` as the regular expression.

  Note: In regular expressions, `.*` matches any sequence of characters.

Important

Abstract Generator does not check the regular expression you enter against any syntax rules. To ensure that you construct the regular expressions correctly, see Appendix D, “Using Regular Expressions in Abstract Generator”.

Specifying Pin Names

You specify the pin names in the *Power pin names*, *Ground pin names*, *Clock pin names*, *Analog pin names*, and *Output pin names* fields. Abstract Generator identifies the respective pin names based on the label strings in the abstract.

You can specify a list of possible names for the pins in the form of regular expressions, each separated by a space, as shown in the figure below.

By default, the *Clock pin names*, *Analog pin names*, and *Output pin names* fields are empty.

When Abstract Generator encounters a label that matches one of the regular expressions, it sets properties as given in the table below. In addition, Abstract Generator adds attributes for the pin in question to the LEF definition.

Pin	Terminal Direction	Terminal Type	Attribute Added for LEF PIN in the LEF File
Power pin	`inputOutput`	`power`	`USE POWER`
Ground pin	`inputOutput`	`ground`	`USE GROUND`
Clock pin	`input`	`clock`	`USE CLOCK`
Analog pin	`input`	`analog`	`USE ANALOG`

© 2020 All Rights Reserved.

Output pin	`output`	`signal`	`USE OUTPUT`

If there is a logical view present for the cell being processed, Abstract Generator checks whether a given terminal has a matching terminal in the logical view. If it does, the terminal direction is adopted from the terminal in the logical view. Otherwise, Abstract Generator uses the value specified in the *Output pin names* field to determine the output pins.

Excluding Existing Terminals

In the *Exclude existing terminals* field, you can specify a list of existing terminals that you want Abstract Generator to ignore. You can specify the terminal names in the form of regular expressions and separate the terminal names in the list by using blank spaces. To ignore all the terminal names, type `.*`.

Abstract Generator will remove the specified terminals. In addition, Abstract Generator will create the deleted terminals afresh from the equivalent text labels existing in the layout. Abstract Generator will issue the warning message ABS-12022 if a deleted terminal is not created from a text label existing in the layout.

Note: The new terminal created by Abstract Generator will not inherit any attributes or properties from the equivalent deleted terminal.

By default, the *Exclude existing terminals* field is empty.

Excluding Existing Pins

In the *Exclude existing pins on layers* field, you can specify a list of layers on which you want Abstract Generator to ignore the existing pins. You can specify the layer names as a list by separating the individual layer names using blank spaces.

Abstract Generator will remove all the pins on the specified layers. If Abstract Generator comes across terminals that do not have any pins, it will issue the warning message ABS-12027 and remove those terminals. Next, Abstract Generator will recreate the deleted pins and any of the deleted terminals afresh from the equivalent text labels existing in the layout.

Note: The new pins (and terminals) created by Abstract Generator will not inherit any attributes or properties from the equivalent deleted pins.

By default, the *Exclude existing pins on layers* field is blank.

© 2020 All Rights Reserved.

Text Tab

To access the *Text* tab, select *Flow – Pins*. Then, click the *Text* tab in the *Running step Pins* form.

The text tab allows you to change the name of the pins created in the Pins view from the original value in the layout. For example, the expression "`\\.extra.*`" would create a pin of VDD from a text label of VDD.extra1. by specifying the appropriate rule on the *Text* tab.

Specifying the Search Depth

The following fields on the *Text* tab allow you to specify the search depth for labels and geometries:

- *Label search depth*: Controls how far down the design hierarchy does Abstract Generator search for text labels during text-to-pin mapping. The default value is `0`.

  This means that Abstract Generator only looks at text labels in the top level of the hierarchy and so only those labels could be used to create starting pin shapes for the extractor.

  Sometimes, the layout can be created with labels deeper in the hierarchy. If this is the case, you can change this setting accordingly. It is, however, recommended that you do

© 2020 All Rights Reserved.

not change this value unless absolutely necessary. There are often many labels at lower hierarchy levels that are *not* intended to become pins in the top level of the abstract.

- *Geometry search depth*: Controls how far down the design hierarchy does Abstract Generator search for metal underneath a text label. The default value is `20`.

  However, if you know that the metal shapes that you want to map to are located at the top of the hierarchy, you can set the option appropriately. This should speed up processing time in the Pins step because only top level shapes will be stored in the geometry cache.

Manipulating Text Labels

Text labels often include special text at the end of the label that is not required for pin naming. Use the *Text Manipulation* table to ensure that the pin names created during the Pins step match the logical pin names. The function uses regular expressions to locate unnecessary text in labels and either removes it completely or replaces it with the text you require.

In the *Text Manipulation* table, you specify the expression to be replaced in the *Remove Text* column and the text to replace with in the *Replace With* column. The use of these columns is further described below.

- *Remove Text*

  Use this field to specify a regular expression representing the pin label text that you are searching.

  - `\\.extra.*`

    Specifies that "`.extra.`" followed by any other character(s) (represented by `*`) is to be removed. For example, `VDD.extra1`, `VDD.extra2... VDD.extra20`, and so on are all converted to `VDD` if the *Replace With* field is left blank.

  - `:.`

    Specifies that "`:.`" followed by any other character(s) is to be removed. For example, `VDD:P` and `VDD:Fred` are both converted to `VDD` if the *Replace With* field is left blank.

  - `\(([0-9]+)\)` and `\[([0-9]+)\]`

    Specifies that the characters `(` and `[` are to be converted to < and the characters `)` and `]` are to be converted to >.

    The *Replace With* field displays `<\1>` in both cases, where `\1` indicates that the *Replace With* text is the first expression matched in parentheses.

© 2020 All Rights Reserved.

- *Replace With*

  Contains the replacement text for the *Remove Text* option. The replacement regular expression text is applied to all matching instances.

The following buttons below the *Text Manipulation* table help to work with the table:

- *Add*

  Adds a new row to the table.

- *Remove*

  Deletes the selected rows from the table.

Example

To change parenthesis, `()`, to angle brackets, `<>`, for the label `(Y)`, enter the following in the *Text Manipulation* fields.

Remove Text...	Replace with...	Result
`\(`	`<`	`<Y)`
`\)`	`>`	`<Y>`

Generating Additional Pins from Labels

Set the `PinsFromTextForExistingPins` option to true to generate additional pins from labels, even if the corresponding net has existing pins in layout. For information about how to set this option, see `PinsFromTextForExistingPins`.

Preserving Text Labels from the Layout View

If you select the *Preserve text labels* option in the *Running step Pins* form, all existing pins in the layout, along with their labels, irrespective of whether the pins were created by Abstract Generator, are retained during the Abstract step. This option is selected by default.

Alternatively, you can use the `PinsTextPreserveLabels` option to preserve pins and their labels.

```
absSetBinOption("Core" "PinsTextPreserveLabels""true")
```


© 2020 All Rights Reserved.

Ensure that the layer of the text label is included in `PinsTextPinMap`. For more information about `PinsTextPinMap`, see Pin Annotation.

Restricting Pins to the PR Boundary

The state of the *Restrict Pins to PR Boundary* option controls the geometry of N-Well pins created during the abstract generation process. If selected, the size of N-Well pins is restricted to the PRBoundary. For more information, see Creating N-Well and P-Well Pins.

© 2020 All Rights Reserved.

Boundary Tab

To access the *Boundary* tab, select *Flow – Pins*, then click the *Boundary* tab in the *Running step Pins* form.

The options in the *Boundary* tab control the calculation of the place-and-route (PR) boundary, which defines the size of a single cell or block.

In addition to specifying whether or not the PR boundary is to be created, you can specify the layers to be used for calculating the PR boundary and the details for adjusting or fixing the size of the calculated boundary.

Specifying the Creation of the PR Boundary

You can specify whether Abstract Generator is to create a PR boundary by choosing one of the values from *always*, *as needed*, and *off* from the *Create boundary* cyclic field. The new boundary is created as an OA `prBoundary` object. When checking for the existence of a PR boundary, Abstract Generator looks for a PR boundary object. If the PR boundary object is not found, then Abstract Generator searches for the shapes on layer `prBoundary` and purpose `boundary`.

© 2020 All Rights Reserved.

- *always*: With this option, Abstract Generator always creates a PR boundary after deleting any already existing PR boundary. For this, it looks for PR boundary object, deletes it, and then creates the new PR boundary object.
- *as needed*: With this option, Abstract Generator first checks whether a PR boundary exists. It creates a PR boundary only if no PR boundary already exists, for which it searches for PR boundary object in layout.
- *off*: In this case, Abstract Generator never creates a PR boundary, though it checks whether or not a PR boundary exists and reports an error if there is no PR boundary in the layout (unless the *Fix Boundary To* options are specified).

Abstract Generator creates a rectangular PR boundary in the case of both *always* and *as needed* options. To obtain a rectilinear PR boundary, you can use the `PinsCreatePolyPRB` option. For more details about using this option, see Abstract Generator SKILL Functions.

Note: Abstract Generator creates an OA prBoundary during the Pins step itself and propagates it during the Extract and Abstract step.

Place and Route Boundary Shapes with the placementClass Property

PR boundary shapes, which have the `placementClass` property set, are considered by Abstract Generator to be *placement regions*. The `placementClass` property indicates that instances with a matching `placementClass` can be placed in the same placement regions.

Abstract Generator ignores placement regions while determining whether a PR boundary exists. To enable Abstract Generator to correctly recognize a PR boundary, the instance must not have the `placementClass` property set.

Specifying the Layers for PR Boundary Calculation

You can use the *Using geometry on layers* field to specify the layers to be used to calculate the PR boundary. The boundary is drawn so that it encloses all the geometry found on these layers.

For example,

```
metal1 (metal2 drawing)
```

This specifies all geometry on `metal1` all purposes except the ignored purposes and on `metal2` purpose `drawing` to be considered for PR boundary calculation.

© 2020 All Rights Reserved.

Manipulating the Dimensions of the PR Boundary

The *Adjust Boundary By* section on the *Boundary* tab lets you increase or decrease the size of an existing or calculated PR boundary on one or more sides. To do this, specify the distance to be added to each boundary edge.

For example, if you specify a positive number for the *Left* edge, the left boundary will be moved further to the left by that amount, that is, the lower left coordinate of the boundary will be decreased.

The *Fix Boundary To* section lets you fix individual cell boundary edges to an absolute value. These values are applied to existing and calculated PR boundaries.

For example, you have two cells in a bin one of which already has a boundary and the other does not. If you set the *Create boundary* option to *as needed* and specify `-15` in the *Bottom* field of the *Fix Boundary To* section, Abstract Generator will automatically calculate the boundary for the other cell and set the lower edge of both cells to `-15` µ.

© 2020 All Rights Reserved.

Blocks Tab

To access the *Blocks* tab, select *Flow – Pins*, then click the *Blocks* tab in the *Running step Pins* form. The options on the *Blocks* tab are intended for use with any layout view that has existing routing. Typically, these will be routed blocks imported from DEF.

Preserving Local Blockages in Routed Blocks

The *Preserve local blockages in routed blocks* option controls how Abstract Generator processes local blockages defined in the `BLOCKAGE` section in DEF 5.8. Local blockages do not have the `+PUSHDOWN`, `+FILL`, or `+SLOT` modifiers.

- When unchecked, Abstract Generator removes any local blockages and proceeds as if they had never been specified in the DEF file.
- When checked, Abstract Generator preserves the local blockages along with their attributes for all the layers. The blockages along with their attributes are preserved from level 0 to level 32. The preserved layout blockages exists in the abstract view along with the other blockages generated in the abstract step.

  Note: The layout blockages are not merged with the blockages generated in the abstract step and are separately maintained. This is done to enhance the performance.

Creating Power Pins from Routing

The *Create power pins from routing* option controls whether pins are created from the routing defined in the `SPECIALNETS` section of the DEF file.

© 2020 All Rights Reserved.

- When unchecked, Abstract Generator creates pins only from the pin geometry defined in the `PINS` section of the DEF file. If there is no pin geometry, the routing is turned into pins instead.
- When checked, Abstract Generator creates pins from the routing defined in the `SPECIALNETS` section for the layers specified in the *Using geometry on layers* field. It does this even if there are pins already defined in the `PINS` section.

When creating pins, Abstract Generator ignores `FILLWIRE` and `SHIELD` net segments. It also creates pins in instances that are logically connected to the top-level nets.

© 2020 All Rights Reserved.

Creating N-Well and P-Well Pins

Abstract Generator supports the creation of N-Well and P-Well pins. During abstract generation, the tool recognizes the text labels for N-Well and P-Well shapes on the layout and creates corresponding physical pins.

N-Well Pins

To create N-Well pins, during the Pins step, you need to:

- Define the geometric specification of N-Well shapes.

  Example: (Nwell (Nwell drawing))

- Create text labels for the N-Well pins.

During abstract generation, the tool identifies all text labels on the layout design. Then, it recognizes the shapes overlapping each text label and creates pins of the same size as the overlapping shape.

Example:

Note: If there is no Nwell shape overlapping the label, then no pin is created.

For some designs, the N-Well shape boundary may exceed the PRBoundary. By default, Abstract Generator will create N-Well pins of the same size as the N-Well shape. The `PinsRestrictToPRBndry` option enables you to limit the N-Well pin size to the PRBoundary. The default value is false.

© 2020 All Rights Reserved.

P-Well Pins

To create P-Well pins, during the Pins step, you need to:

- Define the geometric specification of the P-Well shapes.

  Example: (Pwell (Pwell drawing))

- Create text labels for the P-Well pins.

During abstract generation, the tool identifies all text labels on the layout design. Then, it recognizes the shapes overlapping each text label and creates P-Well pins of the size of the overlapping P-Well shapes.

© 2020 All Rights Reserved.

If there are no Pwell shapes overlapping a text label, then the size of the P-Well pin is calculated as follows:

Area (P-Well Pin) = Area (PRBoundary) – Area (N-Well Pin)

The Pwell pin is placed adjoining the Nwell pin.

© 2020
All Rights Reserved.

Creating Multiple N-Well and P-Well Pins

To create multiple N-Well or P-Well pins, define the geometric specification of the shapes and then create text labels for the pins.

Examples:

Defining the geometric specification of the N-Well shapes: (Nwell (Nwell drawing))

Defining the geometric specification of the P-Well shapes: (Pwell (Pwell drawing))

If there are no P-Well shapes under the P-Well label, then the P-Well pin are is calculated using the following formula:

Pwellpin = TotalPrBoundary - NwellPin

© 2020
All Rights Reserved.

If an N-Well shape cuts the PrBoundary into multiple regions, then multiple P-Well pins are created.

P-well pins are created only in the regions that have the P-well labels. Notice that a P-Well pin is not created in the middle region of the above example.

If there are multiple P-Well labels at the same region, then a P-Well pin is created only for the first label, and a warning is displayed for the remaining labels.

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

4

Extract Step

The Extract step, the second stage in the abstract generation process, traces the connectivity between shapes and terminals, starting at the *pin* purpose shapes created in the Pins step. The shapes created during extraction are created on the purpose *net* at the top level of the extract view.

Abstract Generator performs the following functions during the Extract step:

- Extracts each terminal net, one shape at a time
- Constructs the correct database model for strong, weak, and must-join pins
- Creates library process antenna information for custom blocks and standard cells

To run the Extract step, select one or more cells in the main window and click the *Extract* button. Alternatively, select *Flow – Extract*. This opens the *Running step Extract* form. You can select the options you need and click the *Run* button. The *Percent Complete* message box reports the progress of the step.

If you have not already generated pins for the selected cells, Abstract Generator automatically runs the Pins step prior to running the Extract step. For information about pin generation, see Pins Step.

Important

You can enable or disable extraction for signal and power nets or specify the extraction to be solely performed for antenna data. The default settings depend on a cell's bin type, and by default, all bins except the *Core* bin have extraction turned off.

When the Extract step is complete, Abstract Generator places a graphical icon against the cell to indicate the status of the Extract view that was created. For information about these icons, see Viewing Status Information.

Note: Abstract Generator automatically detects shapes placed in WSP regions and ignores these shapes during net extraction. Therefore, no pins are created for such shapes.

© 2020 All Rights Reserved.

This chapter describes the following tabs on the *Running step Extract* form:

- Signal Tab
- Power Tab
- Antenna Tab
- General Tab

In the *Running step Extract* form, use the radio buttons on the left to navigate to the tab pages for the different steps and bins you are using.

Some of the common tasks that you need to perform for signal and power net extraction and antenna data calculation include:

- Specifying Layers and Geometries for Extraction
- Specifying the Extraction Limits
- Excluding Metal Purpose Blockages in Layout Designs

You do not need to specify extraction limits for antenna data calculation.

For information about the other abstract generation steps, see the following:

- Pins Step
- Abstract Step

© 2020 All Rights Reserved.

Specifying Layers and Geometries for Extraction

To extract geometries on signal or power nets or to calculate process antenna information, you need to provide information about them to Abstract Generator. The Extract step provides three different tab locations - *Signal*, *Power*, and *Antenna* - for you to provide these details. On all these tabs, you will be required to specify the geometry specification details in the respective layer assignment table.

The layer assignment table typically comprises the following columns:

- Layer
- Geometry Specification
- Create Pins

Layer

In this column, you can select from a list of all the layers specified in the technology file through which you want to extract signal or power nets or antenna data. Each row in this column provides a drop-down list of the layers in the technology file.

You can use this column to limit the extraction to shapes on a subset of all the layers. For example, if you want to prevent Abstract Generator from extracting through via layers, you can delete those layers from this list.

Geometry Specification

For each layer included in the *Layer* column of the table, enter a layer-geometry specification that you want to use for extraction.

For example, if you want to extract through diffusion layers you will usually want to preprocess the diffusion geometry by subtracting the polysilicon layer geometry first. You can achieve this by specifying the following geometry specification for the diffusion layer:

```
diff andnot polysilicon
```

The first entry in the expression is the layer name. The second entry is a more complex expression combining as many layer names and layer operators as required.

Note: If you specify only a layer name instead of a layer-purpose pair in the *Geometry Specification* column, Abstract Generator considers all the purposes on that layer except the Ignored Purpose(s).

© 2020 All Rights Reserved.

Tip

Use the G-Spec Builder Form to help you with the geometry specification. To access this form, place the cursor in the *GeometrySpecification* field for the required layer and click the Edit button.

You can also use this column to specify more than just the list of layers to be extracted. Rather than listing just the layer name for each layer, you can enter an expression that is used to specify how the input geometry for the extraction of that layer is derived.

Note: If you attempt to define two geometry specifications for the same layer, an error will be issued.

Create Pins

You select the *Create Pins* option corresponding to layers in the *Layers* column on which you want pins to be created as a subset of the layers used in the extraction. This helps to determine the shapes found by the extractor that are converted into pins. By default, the *Create Pins* option is active when a new layer is added to the table.

Working with the Layer Assignment Tables

Typically, the layer assignment tables are accompanied with three buttons—*Add*, *Edit*, *Delete*—to enable you to work with the table. In addition, the G-Spec Builder Form enables you to construct expressions for the geometry specifications.

Add

Adds a new layer row to the table.

Edit

Opens the G-Spec Builder Form for the selected layer.

Delete

Deletes the selected layer rows. Check the box to the left of a row to mark it for deletion.

© 2020 All Rights Reserved.

G-Spec Builder Form

Allows less experienced users to build their own geometry specifications (g-specs) using temporary layers. These temporary layers are then translated into the correct g-spec syntax when the *OK* button on the form is clicked.

To add a new blank row in the *G-Spec Builder* form, click the *Add* button. Click *Delete* to remove any rows selected for deletion. Selecting the check box at the far left of each row marks a row for deletion.

You can create a g-spec only for one layer at a time. If you want to create g-specs for multiple layers, you must enter the first g-spec you have specified into the original table (by clicking the *OK* button), click in the *Geometry Specification* field of the next layer you want to create a g-spec for, then click the *Edit* button again to reopen the *G-Spec Builder* form.

For g-specs that are more complex than two layer-purpose pairs and an operator, it is necessary to use temporary layers. To do this, you must first define a temporary layer to be the equivalent of a g-spec. Its definition can then be used at any future time within the *G-Spec Builder* form. When it is translated into proper syntax, the temporary layer definition is substituted.

© 2020 All Rights Reserved.

For example, consider the following entry in the *G-Spec Builder* form.

This g-spec would translate into the *Layer* and *Geometry Specification* fields in the main layer assignment table as described below.

Layer	Geometry Specification
`METAL1`	`((METAL1 pin or METAL1 net) size 0.5) and METAL1 drawing)`

The table columns on the *G-Spec Builder* form are described below.

Result	This is the layer on which the resulting shapes from the Boolean operation will be placed. You can do one of the following: ■ Select a layer that exists in the technology file. ■ Manually enter the name of a layer that exists in the technology file. ■ Manually enter the name of a temporary layer.

© 2020 All Rights Reserved.

Layer 1	This is the layer of the layer-purpose pair being used for defining the geometry specification. You can do one of the following: ■ Select a layer that exists in the technology file. ■ Manually enter the name of a layer that exists in the technology file. ■ Manually enter the name of a temporary layer.
Purpose 1	This is an al field. As above, the layer or layer-purpose pair must be present in the technology file.
Operator	Select an operator to use in the Boolean operation. The `size` operator accepts a float number rather than another layer-purpose pair as an argument. In this case, the layer field already defined in the technology file is editable. For more information, see Operators.
Layer 2	This is the layer of the second layer-purpose pair being used for defining the geometry specification. You can do one of the following: ■ Select a layer that exists in the technology file. ■ Manually enter the name of a layer that exists in the technology file. ■ Manually enter the name of a temporary layer.
Purpose 2	This is an al field where you can specify the name of a purpose. The layer or the layer-purpose pair must be present in the technology file.

Caution

Take care while creating g-specs to act on pins on the `prBoundary` layer. This layer can contain many thousands of shapes and an incorrect g-spec can cause them to merge into a single rectangle for each pin.

© 2020 All Rights Reserved.

Operators

The following is a list of the Boolean operators that can be used in the G-Spec Builder Form.

geomSpec	=	*layer* | *layer-purpose* | *geomSpec* `binaryOp` *geomSpec* | *geomSpec* `size` *float* | *geomSpec* `holes` | `noholes` | *(geomSpec)*
`binaryOp`	=	`andnot` | `and` | `or` | `xor` | `inside` | `outside` | `straddle` | `size`

Examples

```
M1
(M1 or M1 pin)
(M1 size 0.5)
(M1 noholes)
```

The operators are described in detail below.

`holes`	A hole is an area created when the perimeter of a polygon touches itself, enclosing an area that is not the polygon. These are sometimes referred to as donut holes. The `holes` operator generates shapes consisting of all the holes in shapes of the input layer.
`noholes`	The `noholes` operator generates new shapes consisting of all the original shapes of the input layer without the holes.
`andnot`	The `andnot` operator generates new shapes from areas of the first input layer that do not overlap any area of the second input layer.
`and`	The `and` operator generates new shapes from the overlap of original shapes on the input layers.
`or`	The `or` operator merges all shape layers. For polygon layers, a merge generates polygons whose area encompasses all the areas from all the layers and removes all the edges internal to the polygons.
`xor`	The `xor` operator generates new shapes from those portions of either input layer that do not overlap the other layer.

© 2020 All Rights Reserved.

`inside`	The `inside` operator selects the shapes on the first layer that are totally inside shapes of the second layer. A first-layer shape is inside if its entire area is covered by the area of a second-layer shape. Coincident shapes are considered inside in this check. If shapes on two layers are coincident on all sides, each layer can be defined as being inside the other.
`outside`	The `outside` operator selects the shapes on the first layer that are totally outside shapes of the second layer. A first layer shape is outside if none of its area is covered by the area of a second layer shape. Abutting edges are considered outside.
`straddle`	The `straddle` operator selects shapes on the first layer that straddle shapes on the second layer. A first-layer shape is straddling if part of its area is covered by a second layer and part is not. Abutting and coincident edges do not affect this function.
`size`	The `size` operator “grows” the shapes on the input layer by the specified amount. If the size amount is negative, the shapes will be shrunk by the specified amount. The resultant shapes are automatically merged.

© 2020 All Rights Reserved.

Specifying the Extraction Limits

You can speed up the extraction process by limiting how far from the starting shape and how far down the hierarchy should Abstract Generator search for pins.

Specifying the Depth

In the *Maximum depth* field, you specify how far down the instance hierarchy should Abstract Generator search for shapes for extraction. The default value for this field is `32`, which is the maximum depth of hierarchy levels in the database.

Limiting the hierarchy search depth is useful if you know the likely location of the shapes to be extracted. For example, if you know that all the geometry that needs to be converted into pins is at the top level (in a block most likely), you can set the value of the *Maximum depth* field to `0`. This value will tell Abstract Generator to look for shapes only at the top level of the hierarchy and not search further, in turn, enhancing the speed of the Extract step. This might be the case with signal pin geometry that might exist on different levels of the design hierarchy, but is useful for connecting to only at the top level where it would typically be at its widest.

Note: You may want to restrict the level depth to extract through to prevent pins from being created on lower levels of the instance hierarchy. However, if you specify a lower value for the *Maximum depth* field and have also selected to perform antenna extraction (see *Layer Assignment for Antenna Extraction*), extraction will be performed twice, first for signal nets and then for antenna data.

Specifying the Distance

The *Maximum distance* field allows you to restrict the shapes considered by the extractor to those that intersect a bounding box around each starting pin shape. The extractor will only extract shapes that intersect the bounding box. The size of this bounding box is specified in microns in the *Maximum distance* field.

This specification can help to reduce the extraction time for large blocks, where you want to restrict the pin geometry to be near the block boundary. For example, when extracting signal

© 2020 All Rights Reserved.

or power nets in a large block, you would supply a maximum extract distance to prevent extracting the entire signal/ground net or power/ground net within the block.

Specifying the Width

In the *Minimum width* field, you specify the minimum width of the shapes to be extracted. The extractor extracts only shapes that are greater than or equal to the width you specify.

The minimum width value is applied to all layers except cut layers. The minimum width criteria is *not* applied to cut layers because the extractor would always stop at the narrow cut shapes.

The benefit of using this option is that it can significantly reduce the time taken to extract useful geometry from large nets that have a number of routing widths.

For example, if you are interested in extracting ring pin geometry from a block that has internal power rings, you can specify the minimum width for extracting power nets to be equal to the width of the ring pins. The extractor would then only extract the ring pin geometry.

Ignoring Cells at the Top Level

Extremely detailed and complicated layout designs might take too much time for extraction. To reduce the extraction time of such layout designs, you can selectively stop the extraction of certain cells at the top level.

Use the `absSetOption` option to specify the cells to be ignored during extraction. Specify the cell names to be ignored as a space separated string in this option. For example, to ignore all instances of `"CELL_55"` and `"CELL_56"` during extraction, use:

```
absSetOption("IgnoreCellsForExtraction" "CELL_55 CELL_56")
```

Note: The default state of this option is:

```
absSetOption("IgnoreCellsForExtraction" "")
```

For ignoring fluid guard rings and Multi-Part Paths (MPPs) that are implemented as Pcell instances, you can set the corresponding cell names in this option. Consequently, all instances of the specified cells are ignored during extraction.

Excluding Metal Purpose Blockages in Layout Designs

A layout design containing shapes on a metal purpose blockage might cause problems during extraction unless you modify the geometry specification settings as are described below. This

© 2020 All Rights Reserved.

is because these shapes usually cover large areas of a design so that when extraction is performed through these layers, all the geometry under the blockage is also extracted. This would obviously impact extraction performance and could cause shorts.

To avoid this potential problem, you can specify the purposes to be ignored in the Ignored Purpose(s) field. By default, shapes on the boundary purpose are ignored. If you want to include the shapes on the ignored purposes during the Pin, Extract, and Abstract steps, you can explicitly mention the layer and the purpose names in the geometry specification at the appropriate locations in the flow steps.

© 2020 All Rights Reserved.

Signal Tab

To access the *Signal* tab, select *Flow – Extract*. The *Signal* tab is the first tab in the *Running step Extract* form.

The *Signal* tab allows you to view and modify the options for extracting signal nets. You can specify the layers through which you want Abstract Generator to extract, the layers on which pins are to be created, the limits of extraction, and the way in which must-join relationships are to be handled.

© 2020 All Rights Reserved.

Relationship between Signal and Antenna Tabs

It is important to understand how options on different tabs—*Signal* and *Antenna*—are interrelated and impact the overall extraction.

The *Antenna* tab in the *Running step Extract* form provides various *Calculate* options for calculating the input, output, and tristate antenna pins and the antenna metal and side areas. If any one of the *Calculate* options is selected, full extraction is performed for establishing antenna properties.

You must specify the polysilicon, contact cut, and diffusion layers, along with a suitable expression for the diffusion layer, such as `(diffn (diffn andnot poly))`, in the Layer Assignment for Signal Extraction table. If you do not provide this information, the extraction process will not be able to locate any polysilicon or diffusion geometry that is required to perform the subsequent antenna calculation and gate/drain recognition. It is not necessary to provide a complex expression for the polysilicon layer; it has to be listed as an extract layer. However, you should ensure that diffusion, contact, or polysilicon layer information is not entered in the pin layer field on the *Map* tab because you only want to extract through these layers and not create pins on them.

Note: It is feasible to turn off the *Extract signal nets* option at this point. This would result in Abstract Generator performing a full extraction in the background to calculate the antenna properties specified in the *Antenna* tab. However, the result would be that the pins created would be the same as those created after the Pins step, that is, no new pins would be created. Effectively, in this scenario, a “behind-the-scenes” full extraction is being performed solely for the purpose of calculating antenna numbers. Alternatively, if the *Extract signal nets* option is turned on, full extraction will be performed and the geometry found during the extraction will be turned into pin geometry.

The table below shows the type of extraction that Abstract Generator would perform based on the options selected on the *Signal* and *Antenna* tabs in the Extract step.

"Extract signal nets" option on the Signal Tab	Any "Calculate" option on the Antenna Tab	"Use different layer assignments for antenna calculations only" option on the Antenna Tab	Type of Extraction Performed
Off	Off	Off	Preset pin extraction

© 2020 All Rights Reserved.

"Extract signal nets" option on the Signal Tab	Any "Calculate" option on the Antenna Tab	"Use different layer assignments for antenna calculations only" option on the Antenna Tab	Type of Extraction Performed
On	Off	Off	Signal extraction only using the values specified in the Layer Assignment for Signal Extraction table.
Off	On	Off	Antenna extraction only using the values specified in the *Layer Assignment for Signal Extraction* table.
Off	Off	On	No extraction performed
On	On	Off	Antenna and signal net extraction using the values specified in the *Layer Assignment for Signal Extraction* table.
Off	On	On	Antenna extraction only using the values specified in the *Layer Assignment for Antenna Extraction* table.

© 2020 All Rights Reserved.

"Extract signal nets" option on the Signal Tab	Any "Calculate" option on the Antenna Tab	"Use different layer assignments for antenna calculations only" option on the Antenna Tab	Type of Extraction Performed
On	On	On	■ Signal and Antenna Layers are the Same The only extraction performed in this case is that using the values specified in the *Layer Assignment for Signal Extraction* table to calculate both antenna and signal nets extraction. ■ Signal and Antenna Layers are Different Two extractions are performed in this case. Antenna extraction is performed first using the values specified in the *Layer Assignment for Antenna Extraction* table, followed by signal net extraction using the values specified in the *Layer Assignment for Signal Extraction* table.

Note: The only other situation where two types of extractions are possible is when the *Maximum depth* option is set to a value different from the default value *20*. However, even in this situation, the rules mentioned in the table above would apply.

Extracting Signal Nets

Selecting the *Extract signal nets* option directs Abstract Generator to run connectivity extraction for signal nets. Therefore, when this option is turned on, Abstract Generator extracts all the pin shapes created during the Pins step. When the option is turned off,

© 2020 All Rights Reserved.

Abstract Generator copies the pin geometry generated in the Pins step but the pin shapes now have purpose *net* rather than purpose *pin*. This is the default while processing blocks.

Selecting the *Extract signal nets* option activates the Layer Assignment for Signal Extraction table on the *Signal* tab. In this table, you specify the layers through which you want Abstract Generator to extract.

Note: If a pin-short is detected, Abstract Generator displays an error message detailing the cause of the problem. The error might occur as a result of either a bad layer, bad extraction option settings, or perhaps a short between “false” boundary purpose shapes and other real shapes.

Extracting Signal Nets for Blocks and IO Cells

Cadence recommends that you should not use the *Extract signal nets* option for blocks or IO cells unless you are sure you need to. For example, use this option only if you want to perform antenna extraction.

If you do need to extract signal nets, do not create pins on a given layer unless it is necessary. For example, do not create pins on polysilicon, diffusion, or contact layers, except where there are dominant cut-layer spacings.

Tip

You should also turn on the *Create boundary pins* option in the Abstract step to ensure that not all of the extracted geometry is turned into pins. If you do not, Abstract Generator would cut around each pin, which would increase the time taken to generate the blockage model.

Assigning Layers for Signal Extraction

You can use the *Layer Assignment for Signal Extraction* table to specify the geometry or shapes that should be present on a particular layer when signal net extraction is performed. For example, you can restrict `metal1` to be just `metal1` purpose *pin* or you can specify `metal1` to be equal to shapes only where `metal1` purpose *pin* and `metal1` purpose *net* overlap.

- You use this table to specify the Layer on which signal net extraction is to be performed. This table becomes active when you select the *Extract signal nets* option.
- You can also set the Geometry Specification for the layers specified in this table.

© 2020 All Rights Reserved.

- In the *Connectivity* column, you can specify the type of connectivity setting, *Strong* or *Weak*, for a particular layer. The connectivity setting is valid for a layer only if a geometry specification is provided for that layer.

  When extracting signal nets, you can set certain layers as “weak” layers and instruct the extractor to extract through these as well. The shapes extracted from weak layers will be stored in the LEF file as `weakly-connected`.

  The router handles strongly and weakly connected pins differently. Where a LEF `PIN` has two strongly-connected shapes, the router can connect to one of the shapes to form the connection to that pin, then continue routing from the other shape on the same pin without adversely affecting performance.

  However, when the shapes are weakly-connected, it adversely affects the delay of the net. Weakly-connected shapes are written out to the LEF file in separate `PORTS` of the same LEF `PIN`. When a router encounters two weakly-connected shapes within a LEF `PIN`, it does not feedthrough from one shape onto the other; it routes to only one of the shapes. You would typically state that a layer has weak connectivity if it has high resistivity, as is the case with polysilicon layers.

- You can Create Pins corresponding to the layers specified in the *Layers* column.

The Add, Edit, and Delete buttons and the G-Spec Builder Form in conjunction with the Operators are used to work with the *Layer Assignment for Signal Extraction* table.

Important

The entries in the *Layer Assignment for Signal Extraction* table are used by other functions even when the *Extract signal nets* option is switched off. The situations in which this might happen are discussed below.

- In the Extract step, if you have a geometry specification with more than two parts, for example as in,

  ```
  (metal1 andnot marker)
  ```

  Abstract Generator finds shapes that touch or overlap the pin shapes. This can result in the creation of pins that are different from the ones created in the Pins step.

  In the above geometry specification example, it might happen that the design has two nets shorted together on `metal1` and the layout has a `marker` layer that indicates the boundary between the two nets. Therefore, any shape on the `metal1` layer and not the `marker` layer which touches an original pin is turned into a pin.

© 2020 All Rights Reserved.

- In the Abstract step, Abstract Generator tries to stretch the pins. To do this, it uses the geometry specification entered in the table to ensure that it converts only the shapes that satisfy the specification into pins.

Refer to the table in the section Relationship between Signal and Antenna Tabs.

Specifying the Extraction Limits

You can use the *Maximum depth*, *Maximum distance*, and *Minimum width* fields for specifying the extraction limits.

© 2020
All Rights Reserved.

Specifying the Creation of Must-Join Pins

In the *Create Must Join Pins If Required* section on the *Signal* tab, you can control what happens when the extractor finds two disjoint groups of shapes extracted for the same net.

Important

The *Extract signal nets* option should be selected in order to generate the must-join pins. If the *Extract signal nets* option is `OFF`, then the *Create Must Join Pins If Required* options do not have any effect.

In the Pins step, where the same text appears in more than one label, multiple pins are created for the same net. These pins are the starting points for the connectivity extractor.

So, for example, if a cell has a net named `A`, the layout designer might label the net `A` at two different locations in the cell. Abstract Generator will create two starting pins for net `A` and the extractor will extract from both of these starting pins.

The extractor might find that the geometries are connected inside the cell. Sometimes, in such cases, the geometries extracted from the two starting pin shapes remain disjoint. This might happen due to one of the following reasons:

- The pins are connected inside the cell but the extraction performed by Abstract Generator has not found that connectivity. This can happen when you limit the extractor to certain layers or when you specify a *Maximum distance* for extraction or limit the hierarchy search to a *Maximum depth*.
- The pins are disjoint inside the cell, and the layout designer intended that the router should connect to both pins in order to correctly connect net `A`. In LEF, this set of pins are termed as “must-join”.

If the `LEF54_MUSTCONNECTPORTS` property is set on any terminal in the layout, it will be reflected in the abstract view.

© 2020 All Rights Reserved.

The *Always* option and the *Only on terminals named* field gives you control over when to create LEF must-join relationships between disjoint geometries extracted from separate starting pins on the same net.

Always: If this option is selected, Abstract Generator will establish a must-join relationship between all disjoint groups of geometries extracted from separate starting pins on the same net.

A must-join relationship is represented in LEF as must-join terminals. Abstract Generator creates must-join terminals with the same name as the original terminal, but also includes the suffix `_mj`***xxx*** where ***xxx*** is the number of the must-join group used internally to identify all the shapes in the same group. For example, in the scenario described above, Abstract Generator would create a must-join terminal called `A_mj1`.

Only on terminals named: If the *Always* option is not selected, Abstract Generator considers the value(s) specified in the *Only on terminals named* field instead.

In this field, enter a list of terminal names, each separated by a single space. If the extractor finds a terminal name matching one of the names supplied, and if there are disjoint geometries from different starting pins on that net after extraction, Abstract Generator creates a `MUSTJOIN` relationship for that terminal.

This might be useful when you know the terminals expected to have must-joins. Specify their names in this field, and if the extraction process creates other disjoint groups of geometries, they will not be turned into must-join groups and will instead be treated as strongly connected.

Abstract Generator informs you when it creates must-join terminals as the result of an extraction. It also reports when it detects disjoint geometry groups but the option settings did not allow the creation of the must-join terminal.

If there are must-join terminals already defined in the layout, they will be retained in the final abstract irrespective of whether you use the options in the *Create Must Join Pins If Required* section or extract signal nets. Abstract Generator would propagate the must-join terminals only if the pin type (input, output, inout) of the must-join pins is the same; otherwise, Abstract Generator will issue an appropriate error message.

Specifying Must-Join and Must-Connect Relationships through the Options File

You can set the following options in the `abstract.options` file to specify must-join and must-connect relationships for terminals:

`ExtractMustJoinAlways` or `ExtractMustJoinTerminals`

© 2020 All Rights Reserved.

	The default value of `ExtractMustJoinAlways` is `false`. When set to `true`, if Abstract Generator finds disjoint groups of shapes extracted for any nets in the layout, it renames the terminals, and sets the `mustJoin` attribute on these terminals. It corresponds to the *Always* checkbox in the Extract page. Alternatively, list specific terminals with the `ExtractMustJoinTerminals` option. By default, the list is empty. This option corresponds to the *Only on terminals named* field in the Extract page.
`ExtractMustConnectAllPinsAlways` or `ExtractMustConnectTerminals`	
	The default value of `ExtractMustConnectAlways` is `false`. When set to `true`, if Abstract Generator finds two disjoint groups of shapes extracted for any net in the layout, it sets the same terminal name for all the pins under it, and applies the `mustConnectAllPins` attribute on these terminals. Alternatively, list specific terminals with the `ExtractMustConnectTerminals` option. By default, the list is empty. These options do not have corresponding fields in the Abstract Generator graphical user interface. Must-connect options take precedence over the must-join options.

Specifying Must-Join and Must-Connect Relationships through SKILL

Must-join and must-connect options can also be specified using the SKILL function absExtract.

© 2020 All Rights Reserved.

Power Tab

To access the *Power* tab, select *Flow – Extract*, then click the *Power* tab in the *Running step Extract* form.

The *Power* tab allows you to view and modify the options for extracting power nets. You can specify the layers through which you want Abstract Generator to extract, the layers on which pins are to be created, and the limits of extraction.

© 2020 All Rights Reserved.

Extracting Power Nets

Selecting the *Extract power nets* option directs Abstract Generator to run connectivity extraction for power nets. Therefore, when this option is turned on, Abstract Generator extracts all the pin shapes created during the Pins step. This option is on by default for standard cells. When the option is turned off, Abstract Generator copies the pin geometry generated in the Pins step but the pin shapes now have purpose *net* rather than purpose *pin*. This is the default while processing blocks.

Important

When processing a custom block in which ring style power pins are required, it is necessary to keep the *Extract power nets* option on. This will ensure that enough power geometry is extracted that can be processed by the ring pin identification algorithm in the Abstract step to identify the shapes to be turned into ring pins.

Selecting the *Extract power nets* option activates the *Layer Assignment for Power Extraction* table on the *Power* tab. In this table, you specify the layers through which you want Abstract Generator to extract.

Extracting Signal or Power Nets for Blocks and IO Cells

Cadence recommends that you should not use the *Extract power nets* option for blocks or IO cells unless you are sure that you need to; for example, use this option only if you want to create ring pins.

If you do need to extract power nets, do not create pins on a given layer unless it is necessary. For example, do not create pins on polysilicon, diffusion or contact layers, except where there are dominant cut-layer spacings.

Tip

You should also turn on the *Create ring pins* or *Create boundary pins* options in the Abstract step to ensure that not all of the extracted geometry is turned into pins. If you do not, Abstract Generator must cut around each pin, which would increase the time taken to generate the blockage model.

Assigning Layers for Power Extraction

You can use the *Layer Assignment for Power Extraction* table to specify the geometry or shapes that should be present on a particular layer when power net extraction is performed.

© 2020 All Rights Reserved.

- You use this table to specify the Layer on which power net extraction is to be performed. This table becomes active when you select the *Extract power nets* option.
- You can also set the Geometry Specification for the layers specified in this table.
- You can Create Pins corresponding to the layers specified in the *Layers* column.

  Power net extraction for large blocks can often be slow if pins are created on via layers. This is partly due to the large number of via pins that can be created. If power net extraction takes too long, abort the run and reextract. This time, however, do not create pins on via layers.

The Add, Edit, and Delete buttons and the G-Spec Builder Form in conjunction with the Operators are used to work with the *Layer Assignment for Signal Extraction* table.

Important

The entries in the *Layer Assignment for Power Extraction* table are used by other functions even when the *Extract power nets* option is switched off. The situations in which this might happen are discussed below.

- In the Extract step, if you have a geometry specification with more than two parts, for example as in,

  ```
  (metal1 andnot marker)
  ```

  Abstract Generator finds shapes that touch or overlap the pin shapes. This can result in the creation of pins that are different from the ones created in the Pins step.

  In the above geometry specification example, it might happen that the design has two nets shorted together on `metal1` and the layout has a `marker` layer that indicates the boundary between the two nets. Therefore, any shape on the `metal1` layer and not the `marker` layer which touches an original pin is turned into a pin.//

© 2020 All Rights Reserved.

- In the Abstract step, Abstract Generator tries to stretch the pins. To do this, it uses the geometry specification entered in the table to ensure that it converts only the shapes that satisfy the specification into pins.

Refer to the table in the section Relationship between Signal and Antenna Tabs.

Specifying the Extraction Limits

You can use the *Maximum depth*, *Maximum distance*, and *Minimum width* fields for specifying the extraction limits.

Specifying the Creation of Must-Join Pins

The options in the *Create Must Join Pins If Required* section are applicable only to signal nets. For more information, see Specifying the Creation of Must-Join Pins.

© 2020 All Rights Reserved.

Antenna Tab

To access the *Antenna* tab, select *Flow – Extract*, then click the *Antenna* tab in the *Running step Extract* form.

© 2020 All Rights Reserved.

The options on this tab let you create process antenna information for custom blocks and standard cells. By default, all the options on this tab are off.

Antenna Calculation by Abstract Generator

Abstract Generator computes the antenna data for a given layer. The antenna parameters for some of these layers are defined in the LEF technology file. Abstract Generator calculates the following (for metal, cut, and poly layers):

- Antenna gate/diffusion areas
- Antenna metal area/metal side area

Antenna data calculation is complete only after the Abstract step is run, which is the next step in flow after the Extract step. Consequently, the *Export LEF* form always takes antenna data from the abstract view rather than from the extracted view.

The computation of antenna data happens in two steps:

1. The main calculation is performed at the end of the Extract step, where Abstract Generator computes antenna data (assuming that all of the extracted geometry is turned into pins in the abstract).
2. During the Abstract step (at the *Adjust* stage), some of the extracted geometry is not turned into pins (for example, if the *Create boundary pins* options for signal and power nets are active), and the process antenna data is removed from layers lower than the lowest pin layer.

For example, assume that you have an extracted net that has *M3* metal, down to *M2* metal, down to *M1* metal, which then connects to a polysilicon gate. This means that there are three computations of `ANTENNAGATEAREA` for *M1*, *M2*, and *M3* (same values) and a further three for `ANTENNAPARTIALMETALAREA` (different values). If, during the *Adjust* stage, *M3* and *M2* become pin shapes but *M1* becomes an obstruction, the computed `ANTENNAMETALAREA` is removed along with the `ANTENNAGATEAREA` for *M1*.

Note:

- When a local interconnect layer connects to a polysilicon gate, the local interconnect layer is considered for calculating the `ANTENNAPARTIALMETALAREA` and `ANTENNAMAXAREACAR`. The poly layer is considered for calculating `ANTENNAPARTIALMETALAREA` and `ANTENNAMAXAREACAR`.
- Abstract Generator generates antenna numbers for any terminal direction type of a pin, including `input`, `output`, `inout`, `tristate`, `unused`, `switch`, and `jumper`.

© 2020 All Rights Reserved.

- Irrespective of whether the *Extract signal nets* option on the *Signal* tab in the *Extract* step is turned on or off, Abstract Generator generates antenna numbers for a terminal considering all the shapes of that terminal as pin shapes. Also, all the paths from top metal to the gate or drain are considered in calculating the antenna numbers, irrespective of whether the paths contain pin shapes or not.

For more information about process antenna effects, see Appendix C, “Calculating and Fixing Process Antenna Violations” in the *LEF/DEF 5.8 Language Reference*.

FinFET Support in Abstract Generator (ICADVM20.1 Only)

At advanced nodes, in addition to the default antenna data calculation behavior, Abstract Generator allows you to calculate antenna gate are for FinFET devices when the gate area is to be derived from number of fins. A user-defined SKILL hook function `getEffectiveAntennaGateArea` is used for calculating the gate area of the FinFET devices. This SKILL hook function should be defined before running the Extract step.

Syntax:

```
getEffectiveAntennaGateArea (binName gateBBox @rest args)
```

Here, `binName` refers to the bin being used and `gateBBox` is the effective gate area for the specified gate geometry. `@rest args` can include `dbCellViewId`, which is the current database cell view ID in the read mode.

Example:

```
procedure(getEffectiveAntennaGateArea(binName bbox @rest cv)
prog((xa xb ya yb effectiveArea)
      xa = caar(bbox)
      xb = caadr(bbox)
      ya = cadar(bbox)
      yb= cadadr(bbox)

      gateArea = abs((xb-xa)*(yb-ya))
      effectiveArea = gateArea /3
      return(effectiveArea)
);prog
);procedure
```


© 2020 All Rights Reserved.

Updating the Technology File

To enable Abstract Generator to calculate antenna data, you need to update the `antennaModels` section of the `foundry` constraint group. You can specify the thickness value in the `techLayerProperties` subsection of the `layerDefinitions` section if antenna side area calculation is to be performed.

Note: Abstract Generator supports the calculation of process antenna information for LEF versions 5.4 and above. It cannot compute process antenna information for LEF version 5.3.

For general information about `antennaModels`, see the following:

- Appendix F, "Specific Requirements for Antenna Calculation" on page 645.
- "Antenna Model Constraints" in Chapter 2 of the *Virtuoso Technology Data Constraints Reference*.

Performing Process Antenna Calculations

The following options on the *Antenna* tab are used for calculating process antenna data:

- Calculate hierarchical antenna
- *Calculate input pin antenna*
- *Calculate output pin antenna*
- Calculate inout pin antenna
- Calculate antenna metal area
- *Calculate antenna metal side area*
- **Calculation of ANTENNAMAXCUTCAR**: This feature is available through a SKILL function.

Important

OpenAccess supports antenna ratios based on only either side area or non-side area specification in the technology file and not both of them simultaneously. If both side and non-side values are specified in the technology file, the side value overrides the non-side value either when the technology file is updated or when LEF IN is used.

Therefore, even if both *Calculate antenna metal area* and *Calculate antenna metal side area* options are selected, only one of the two antenna metal areas is calculated at any time based on the side and non-side information specified in the technology file. The

© 2020 All Rights Reserved.

Calculate antenna metal area option becomes active when the non-side area value is specified in the `antennaModels` section of the technology file and the *Calculate antenna metal side area* option remains disabled (grayed out). Similarly, the *Calculate antenna metal side area* option becomes active when the side area value is specified in the `antennaModels` section and the *Calculate antenna metal area* option remains disabled. For more information about side and non-side areas, see Appendix C, "Calculating and Fixing Process Antenna Violations" in the *LEF/DEF 5.8 Language Reference*.

Example

As shown in the figure below, because the `antennaModels` section in the technology file contains the non-side area specification, the *Calculate antenna metal area* option is active and can be selected and the *Calculate antenna metal side area* option is disabled.

The table below summarizes the functionality of the antenna calculation options and their default states.

	Description	Default Setting
Calculate hierarchical antenna	Calculates antenna values hierarchically by summing the antenna data on standard abstracts.	Switched off and disabled. Becomes active only when one of the other antenna calculation options is selected.
Calculate input pin antenna	Calculates either the `ANTENNASIZE` LEF (if you are using LEF 5.3) or the `ANTENNAGATEAREA` attribute (if you are using LEF 5.4 or above) for input pins.	Switched off for standard cells and custom blocks.

© 2020 All Rights Reserved.

	Description	Default Setting
Calculate output pin antenna	Calculates either the `ANTENNASIZE` LEF (if you are using LEF 5.3) or the `ANTENNAGATEAREA` and `ANTENNADIFFAREA` attributes (if you are using LEF 5.4 or above) for output pins and tristate pins.	Switched off for standard cells and custom blocks.
Calculate inout pin antenna	Calculates either the `ANTENNASIZE` LEF (if you are using LEF 5.3) or the `ANTENNAGATEAREA` and `ANTENNADIFFAREA` attributes (if you are using LEF 5.4 or above) for IO pins and tristate pins.	Switched off for standard cells and custom blocks.
Calculate antenna metal area	Calculates the area of metal that would be connected to the pin when that layer is fabricated. This option is designed to be used with blocks. The result is expressed in square microns.	Switched off and disabled. Becomes active only when the non-side area ratio is specified in the technology file.
Calculate antenna metal side area	Calculates the side area of the metal (perimeter x thickness) that would be connected to the pins when that layer is fabricated.	Switched off and disabled. Becomes active only when the side area ratio is specified in the technology file.

Some of the options and the calculations related to them are described below.

- ***Calculate hierarchical antenna***

  This option bypasses the check for *Gate* or *Drain* regions in the *Layer Assignment for Antenna Regions* table because these measurements are not required for hierarchical antenna calculations.

  If this option is selected and there are no abstract instances, Abstract Generator performs a flat antenna calculation instead. If this is the case and the *Gate* and *Drain* regions are not specified, Abstract Generator warns you that neither process antenna information has been calculated nor the diffusion and polysilicon geometries have been extracted.

  Calculation

© 2020 All Rights Reserved.

The following equation is the `MaxAreaCar`, `MaxSideAreaCar` and `MaxCutCar` hierarchical antenna calculation for the abstract instance terminal on a particular net. This is maximized with the `MaxCar`'s for the other gate nodes to get the overall max.

```
M_i CAR = M_i PAR + max{ M_i-1 CAR, (M_i MAX_CAR(H) - M_i PAR(H)) }
```

where,

`M_i` is the fabricated metal layer.

`M_i PAR` is the partial metal area divided by the gate area under the metal layer.

`M_i-1 CAR` is the cumulative area ratio calculated for the layer below.

`M_i MAX_CAR(H)` is the `pcsAntennaMaxAreaCAR` information from the instance terminal.

`M_i PAR(H)` is `pcsAntennaPartialMetalArea` or `pcsAntennaGateArea` from the instance terminal.

If `pcsAntennaMaxAreaCAR` is not specified, the calculation takes a simpler form as in:

```
M_i CAR = M_i PAR + M_i-1 CAR
```

Note: Also see Keywords Used in Antenna Calculation.

■ *Calculate inout pin antenna*

The following example illustrates how the antenna values are calculated for inout pins.

Example

Consider the following standard cell output that has internal feedback to the gate `inp1` and two pins (P) in the abstract `P1` (on `METAL1`) and `P2` (on `METAL2`).

`ANTENNAGATEAREA METAL1` = Area of `inp1` gate
`ANTENNAGATEAREA METAL2` = Area of `inp1` gate

`ANTENNADRAINAREA METAL2` = Area of `out1` drain

© 2020 All Rights Reserved.

`ANTENNAPARTIALMETALAREA METAL1` = Area of `METAL1` segment (1) - area of pin `P1`
`ANTENNAPARTIALMETALAREA METAL2` = Area of `METAL2` segment (2) - area of pin `P2`

`ANTENNAPARTIALMETALSIDEAREA METAL1` = Perimeter of (segment (1) - `P1`) + `METAL1` thickness
`ANTENNAPARTIALMETALSIDEAREA METAL2` = Perimeter of (segment (2) - `P2`) + `METAL2` thickness

Note: Also see Keywords Used in Antenna Calculation.

■ *Calculate antenna metal area*

The calculated antenna metal area is added by the router to the routing metal at the next level up in order to compute whether or not there is an antenna violation on that pin or on any other pin on the same net. Use the `ExtractAntennaIncludePolyCAR` option to include poly shape area when calculating the antenna metal area. When set to `true`, poly shapes are considered as metal shapes and are therefore included in the calculation of `ANTENNAPARTIALMETALAREA`.

Consider the following example that describes the calculation of `ANTENNAPARTIALMETALAREA`.

Example

For each metal layer, you must calculate the area of metal attached to the pin that will be connected when that layer is fabricated, that is, the metal on the same external pin side of some higher level piece of metal. You need to calculate the metal area only for those layers where there is pin geometry and any higher layers used internally on the connection to the gate.

From the above figure,

`ANTENNAGATEAREA METAL1` = Area of gate (`input1`)
`ANTENNAGATEAREA METAL2` = Area of gate (`input1`) + area of gate (`input2`) + area of gate (`input3`)

`ANTENNAPARTIALMETALAREA METAL1` = Area of `METAL1` segment (1) - area of `Block pin P`

© 2020 All Rights Reserved.

`ANTENNAPARTIALMETALAREA METAL2` = Area of `METAL2` segment (2) + area of `METAL2` segment (4)

`ANTENNAPARTIALMETALSIDEAREA METAL1` = Perimeter of (`METAL1` segment (1) - `Block pin P`) x `METAL1` thickness
`ANTENNAPARTIALMETALSIDEAREA METAL2` = (perimeter of `METAL2` segment (2) + perimeter of `METAL2` segment (4)) x `METAL2` thickness

Note: Also see Keywords Used in Antenna Calculation.

© 2020 All Rights Reserved.

Calculation of ANTENNAMAXCUTCAR

If you select the Calculate antenna metal area option, Abstract Generator also calculates ANTENNAMAXCUTCAR. This value is calculated for the cut layer just above the lowest metal layer on which the pin is present.

Calculation

```
Cut_(i,i+1) CAR = Cut_(i,i+1) PAR + max{ Cut_(i-1,i) CAR,
(Cut_(i,i+1) MAX_CAR(H) - Cut_(i+1) PAR(H)) }
```

where,

Cut_(i,i+1)	Cut layer above the fabricated metal layer
Cut_(i,i+1) PAR	Partial cut area divided by gate area under the i'th metal layer
Cut_(i-1,i) CAR	Cumulative area ratio calculated for the cut layer below
Cut_(i,i+1) MAX_CAR(H)	pcsAntennaMaxCutCAR value from instance terminal
Cut_(i,i+1) PAR(H)	pcsAntennaPartialCutArea or pcsAntennaPartialGateArea value from instance terminal

By default, the calculation of ANTENNAMAXCUTCAR does not take the CUT01 layer into account. To enable Abstract Generator to consider CUT01 layer while calculating ANTENNAMAXCUTCAR, use the SKILL option ExtractAntennaMaxCutCARLowestCut. For the option usage, see Abstract Generator SKILL Functions.

© 2020 All Rights Reserved.

Example

Consider `PINA` is on `METAL2` layer. In this case, Abstract Generator will calculate `ANTENNAMAXCUTCAR` for `CUT23` layer as this cut layer is above the lowest metal layer containing the pin. By default, the partial cut areas of `CUT23` and `CUT12` will be considered for `ANTENNAMAXCUTCAR` calculation for `CUT23`. If you set the `ExtractAntennaMaxCutCARLowestCut` option to `true`, `ANTENNAMAXCUTCAR` for `CUT23` calculation will also include the partial cut area of `CUT01`.

```
ANTENNAMAXCUTCAR value CUT23
```

Similarly, if `PINA` is on `METAL1` layer and `ExtractAntennaMaxCutCARLowestCut` is `true`, then `ANTENNAMAXCUTCAR` calculation will be considered for `CUT12` as well as `CUT01` layers.

```
ANTENNAMAXCUTCAR value CUT01
ANTENNAMAXCUTCAR value CUT12
```

Note: If the `ExtractAntennaMaxCutCARLowestCut` option is `true`, `ANTENNAMAXCUTCAR` value is displayed for `CUT01` only if the pin is on `METAL1`.

© 2020 All Rights Reserved.

Keywords Used in Antenna Calculation

The following table describe antenna keywords related to *per layer library data*.

Keyword	Description
`Area`	The area of metal wires connected to the polysilicon gate: width(W) x length(L).
`SideArea gate`	The side wall area of metal wires connected to the polysilicon gate: 2(W+L) x thickness.
`Diff gate`	Top mask layer or any metal layer connecting between polysilicon and output diode.
`Cum polysilicon`	Any lower metal layer that connects directly to the gate(s) is also cumulated to the antenna calculation on the metal layer currently being checked.
`PWL variable`	Piecewise linear format. This is used to define the diffusion antenna ratio.
`PAR`	Partial antenna ratio.
`CAR`	Cumulated antenna ratio.
`PAE`	Process antenna effect.
`DIFFUSEONLY`	Keyword to specify the current antenna factor to only be used when the corresponding layer is connected to diffusion.

Checks Performed During Antenna Calculation

If you activate antenna extraction by selecting any *Calculate* option, Abstract Generator checks for the following:

- Extraction of polysilicon layer
- Extraction of diffusion layer
- Specification of diffusion layer in terms of

  `(Diff (Diff andnot Poly))`

If any of the above checks fail, the following corresponding error messages will be displayed:

- Failure of extraction of polysilicon layer generates the following error message:

© 2020 All Rights Reserved.

```
ABS-606: Cell name: layer POLY is not listed in any of the layer
assignment for extraction tables. This is required for antenna
extraction.
```

- Failure of extraction of diffusion layer generates the following error message:

```
ABS-607: Cell name: layer DIFF is not listed in any of the layer
assignment for extraction tables. This is required for antenna
extraction.
```

- Missing specification of diffusion layer in terms of `(Diff (Diff andnot Poly))`

```
ABS-608: Cell name: layer DIFF must be specified in one of the
layer assignment for extraction tables as type (diff diff andnot
poly) for full antenna extraction to take place.
```

To avoid receiving the above extraction-related error messages, ensure that you have entered a valid expression in the Layer Assignment for Signal Extraction table on the *Signal* tab and *Layer Assignment for Antenna Extraction* table on the *Antenna* tab. For example, you can specify the following in these tables:

Layer	Geometry Specification
Poly1	Poly1
CONT	CONT
NDIFF	NDIFF andnot Poly1

Assigning Layers for Antenna Regions

Layers are assigned to antenna regions for identifying the gate and drain regions and to enable antenna calculation.

The *Layer Assignment for Antenna Regions* table is enabled when at least one of the calculate antenna options is active.

© 2020 All Rights Reserved.

If the information in the technology file is set up correctly (layers need to be defined in the `functions` section of the `LayerRules` section), Abstract Generator automatically adds the entries to the table.

Important

In LEF, polysilicon and diffusion layers are defined as masterslice layers and both are converted to poly layers after the LEF data is imported to a technology file. Distinction between these layers is required for antenna calculation. So, Abstract Generator searches for the layer names `oxide`, `diff`, `ndiff`, `diffn`, `pdif`, `pdiff`, and `diffp` corresponding to the layer type `poly`, marks those layers as diffusion layers in the technology file, and removes them from the `poly` layer list. It then uses this information to automatically populate the *Gate* and *Drain* regions in the *Layer Assignment for Antenna Regions* table.

The columns of the *Layer Assignment for Antenna Regions* table are described below.

Layer

Select the polysilicon and diffusion layers specified in the technology file.

Geometry Specification

Enter a layer-geometry specification defining the gate or drain geometry regions. For example,

- `(polysilicon and (pdiff or ndiff)` defines a gate region
- `(pdiff (pdiff andnot polysilicon)` defines a drain region

© 2020 All Rights Reserved.

Tip

Use the G-Spec Builder Form to help you with the geometry specification. To access this form, place the cursor in the *Geometry Specification* field for the required layer and click the *Edit* button.

Note: You can define only one geometry specification for each layer.

Region

Specify whether the layer is a *Gate* or *Drain* region.

Gate	When calculating the size of a MOS transistor gate by using the option *Calculate input pin antenna*, Abstract Generator must measure the area of the region where polysilicon overlaps diffusion rather than the area of the polysilicon wire itself. Consequently, a geometric operation (`AND`) must be carried out to find the gate geometry. Selecting *Gate* as the region and entering the following in the *Geometry Specification* field specifies this operation: `(poly (poly and (ndiff or pdiff))`

Drain	When using either the *Calculate output pin antenna* or *Calculate inout pin antenna* option, Abstract Generator cannot consider only the area of the drawn diffusion; it needs to remove the drain area from the area of the diffusion.
	
	The following geometry specification calculates the n-diffusion drain by subtracting the polysilicon from the n-diffusion geometry:
	
	`(diffn (diffn andnot poly))`
	
	The same principle applies when calculating p-diffusion.
	
	`(diffp (diffp andnot poly))`

Oxide

The *Oxide* entry offers 32 oxide types (the maximum number handled by LEF): *Oxide1 … Oxide32*. This lets you apply a unique oxide type for up to 32 diffusion layers.

If you list more than four diffusion layers, layers with the same *Oxide* type are treated in the same way. You can also select *Oxide1* for all the listed diffusion layers, in effect replicating the LEF 5.7 or above antenna style model.

© 2020 All Rights Reserved.

If the *Oxide1* keyword is associated with a particular diffusion layer, it means that the gate area over this diffusion area and the `ANTENNAMAXAREACAR`s associated with this gate area use the *Oxide1* antenna ratios to determine antenna violations in Innovus.

An example of LEF 5.5 could contain:

```
LAYER METAL1

    ANTENNAMODEL OXIDE1 ;
    ANTENNAAREARATIO 14 ;
    ANTENNAMODEL OXIDE2 ;
    ANTENNAAREARATIO 18 ;

....

END METAL1
```

and

```
MACRO mycell

    PIN mypin

....

        ANTENNAMODEL OXIDE1 ;
        ANTENNAGATEAREA 8.2 LAYER M1 ;
        ANTENNAMODEL OXIDE2 ;
        ANTENNAGATEAREA   7.2 LAYER M1 ;

....

    END mypin

END mycell
```

Note: The *Oxide* option is disabled if you select *Gate* as the *Region* type for a particular layer. Similarly, if you select an *Oxide* type for a layer, the *Region* type is changed to *Drain*.

The default value " " corresponds to *Oxide1*.

The Add, Edit, and Delete buttons and the G-Spec Builder Form in conjunction with the Operators are used to work with the *Layer Assignment for Antenna Regions* table.

© 2020 All Rights Reserved.

Assigning Layers for Antenna Extraction

You can select the *Use different layer assignments for antenna calculations only* option to set layer assignments for antenna extraction different from those set for signal net extraction set on the Signal Tab. Selecting this option activates the *Layer Assignment for Antenna Extraction* table and consequently over-rides any *Layer Assignment for Signal Extraction* option settings. A “behind-the-scenes” antenna extraction calculation can then be performed.

Important

If you are performing “behind-the-scenes” antenna calculations and using replay scripts, you must change the `extractLayersSig` option in the script to `antennaLayers` for it to run successfully.

Selecting this option can be useful, for example, in such cases where you want to create pins from shapes which are touching the initial pin on the same layer and, if you had to specify the via layers to extract through, the Extract step may create pin shapes on other metal layers which you do not require. This, in turn, could cause electro-migration issues.

By default, this option is disabled and off. It is activated if one or more calculate antenna options are selected. Refer to the table in the section Relationship between Signal and Antenna Tabs.

The *Layer Assignment for Antenna Extraction* table is active when the *Use different layer assignments for antenna calculations only* option is selected.

The columns of this table are described below.

- You use this table to specify the Layer for which you want to extract antenna data.

© 2020 All Rights Reserved.

- Enter a Geometry Specification for each layer through which you want to extract antenna data.

The Add, Edit, and Delete buttons and the G-Spec Builder Form in conjunction with the *Operators* are used to work with the *Layer Assignment for Antenna Extraction* table.

Comparing Geometry Specifications for Antenna Regions and Antenna Extraction

When defining the geometry specification for the polysilicon layer in the Layer Assignment for Antenna Extraction table, do not use the same specification as the one you used in the *Layer Assignment for Antenna Regions* table. If you do, Abstract Generator cannot extract the full polysilicon shape and cannot output antenna values for the input gate.

In the following example, notice the different geometry specifications in the two tables.

Layer Assignment Table	Geometry Specification
Antenna Regions	`(polysilicon and (pdiff or ndiff)`
Antenna Extraction	`poly`

Excluding Signal Nets from Antenna Calculation

The *Signal nets to be excluded from antenna calculation* field allows you to specify the nets that should not be considered for antenna extraction and therefore, lets you avoid antenna calculations for the specified pins. For example, by specifying the expression as `in*` for the signal nets to be excluded, you can exclude all nets with names beginning with `in` from antenna calculation.

This field is enabled only if one of the calculate antenna options is selected.

© 2020 All Rights Reserved.

General Tab

To access the *General* tab, select *Flow – Extract*. Then, click the *General* tab in the *Running step Extract* form.

© 2020
All Rights Reserved.

Specifying the Existence of Connectivity Information in Design

Select the option *Use net information in design* if your design already contains connectivity information. If you select this option, Abstract Generator will skip the extraction based on overlapping shapes and use the existing connectivity information in your design to create nets.

Selecting this option for designs with existing connectivity information results in huge performance gain.

Specifying the Connectivity Between Layers

The *Layer connectivity* field is used to specify the connectivity between layers. This allows you to specify the layer connectivity without having to add via definitions in the technology file.

The default value is however based on the connectivity derived from analyzing the vias in the technology file, for example `(M1 M2 V12);` the `poly` and `diffusion` connections, for example `(POLY M1 CONT) (DIFF M1 CONT);` and `local interconnect` layers.

The format for entering a value in this field is to list the layers as a number of entries; each entry being either a layer pair or layer triple.

Example 1:

```
(layer1 layer2 layer3) (layer4 layer5 layer6) ... (layer7 layer8) ...
```

In the above example, a triple layer entry means that `layer1` connects to `layer2` through the via layer `layer3`. A pair means that the first layer, `layer7`, connects to the second layer, `layer8`, by overlap (for example, local interconnect to polysilicon).

Note: You must also specify these layer connections in the Layer Assignment for Signal Extraction table.

Example 2:

```
(Li1 Li2) (PO Li1) (OD Li2) ...
```

In the above example, layer connectivity is defined between local interconnect, poly, and oxide layers. Here, `Li1` and `Li2` are two local interconnect layers, `PO` is a poly layer, and OD is an oxide layer.

Abstract Generator uses this information during extraction to determine where to extract next. Starting from a given shape, the extractor looks at the connectivity defined for the layer on which that shape is present to determine the layers to look for shapes on next.

© 2020 All Rights Reserved.

Note: Bridge vias are also supported during extraction.

Considerations for Specifying Layer Connectivity

- If there is a `metal0` local interconnect layer, it will have to be entered manually.
- If two layers connect by overlap, specify a layer pair `(M0 M1)`.
- It is not permitted to specify `(layer1 layer2 via12)` and `(layer1 layer2)` and so on at the same time.

For example,

```
(metal1 metal2 via12) (metal2 metal3 via23)
(poly metal1 cont)
(pdiff metal1 cont)
(ndiff metal1 cont)
(poly li)
(ndiff li)
(pdiff li)
```

This means,

`metal1` connects to `metal2` through `via12` layer
`metal2` connects to `metal3` through `via23` layer
`poly` connects to `metal1` through `cont` layer
`pdiff` connects to `metal1` through `cont` layer
`ndiff` connects to `metal1` through `cont` layer
`poly` connects to local interconnect by overlap
`ndiff` connects to local interconnect by overlap
`pdiff` connects to local interconnect by overlap

Assigning Layers for Pin Geometry

The *Pin Geometry Restriction* table provides you with finer control over final pin geometry.

© 2020
All Rights Reserved.

Layer

From a list of all the layers specified in the technology file, select those that you want to assign for pin geometry.

In addition to selecting from the *Layer* list, you can type in the layer you want to assign for pin geometry. However, this layer must still be present in the technology file. If it is not, an invalid layer warning is displayed.

Geometry Specification

Enter the Geometry Specification for the regions where pins are either to be restricted to or avoided.

Restriction

Choose whether pins are to be allowed or avoided (*Allow* or *Avoid*) in the geometry specified in the *Geometry Specification* field. The specified geometry specification (G-spec) describes the region where pins are to be allowed or avoided.

By default, the *Allow* or *Avoid* option is used for both signal and power pins. However, you can choose whether only signal or power pins are to be restricted to the specified region or avoided. To do this:

1. Set the following Shell environment variable before starting the Virtuoso session:

   ```
   setenv AllowAvoidSignalAndPowerPinsSeparately
   ```

2. Set the following options of the `absAbstract` SKILL function to specify only power pins.

   ```
   absSetBinOption("Core" "AbstractAdjustAllowPowerPin" "(Metal1 (Metal1 andnot
   prBoundary boundary size -1)) ")
   absSetBinOption("Core" "AbstractAdjustAvoidPowerPin" "(Metal2 (Metal2 andnot
   prBoundary boundary size -1)) ")
   ```

3. Set the following options of the `absAbstract` SKILL function to specify only signal pins.

   ```
   absSetBinOption("Block" "AbstractAdjustAllowPin" "(Metal1 (Metal1 andnot
   prBoundary boundary size -0.05)) ")
   absSetBinOption("Block" "AbstractAdjustAvoidPin" "(Metal1 (Metal1 andnot
   prBoundary boundary size -0.05)) ")
   ```

If the `AllowAvoidSignalAndPowerPinsSeparately` environment variable is not set, the `AbstractAdjustAllowPin` and `AbstractAdjustAvoidPin` options are used to specify both signal and power pins.

© 2020 All Rights Reserved.

Use the *Allow* or *Avoid* option for adjusting pins and not for deleting the existing pins, which can slow down the performance of abstraction process. To create the required pins, specify the geometry specification for the pin layers during the Extract step. For more information, refer to Specifying Layers and Geometries for Extraction.

Use the Add, Edit, and Delete buttons and the G-Spec Builder Form in conjunction with the Operators to work with the *Layer Assignment for Pin Geometry* table.

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

5

Abstract Step

This chapter describes the third stage of the abstract generation process: the Abstract step. Abstract Generator performs the following functions during the Abstract step:

- Adjusts pin shapes
- Performs blockage modeling
- Creates sites
- Calculates overlap layers

To access the *Running step Abstract* form, select one or more cells in the main window and click the *Abstract* button. Alternatively, select *Flow – Abstract*. This opens the *Running step Abstract* form. You can select the options you need on this form and click the *Run* button. The *Percent Complete* message box reports the progress of the step.

If there are no valid Pins or Extract views for a cell, Abstract Generator automatically runs the Pins and Extract steps prior to running the Abstract step. For information about the Pins and Extract steps, see Pins Step and Extract Step.

Important

The QuickAbstract option enables you to improve the performance of Abstract Generator by 30 percent for small and medium sized designs. If this option is set to true, Abstract Generator does not generate intermediate cellviews namely Pin and Extract cellviews; it generates only Abstract cellviews. By default, the QuickAbstract option is set to false.

When the Abstract step is complete, Abstract Generator places a symbol against the cell in the *Cell* pane to indicate the status of the Abstract view that was created. For information about these icons, see Viewing Status Information.

Note: Abstract Generator automatically detects shapes placed in WSP regions and ignores these shapes while creating blockages.

This chapter describes the following tabs on the *Running step Abstract* form:

© 2020 All Rights Reserved.

- Adjust Tab
- Blockage Tab
- Density Tab
- Fracture Tab
- Site Tab
- Overlap Tab

The chapter also discusses the following:

- Retaining Layout Information in the Abstract View
- Creating Fill Blockages Automatically on Specific Layers
- Bus Annotation in Abstract Generator
- Supporting EXCEPTPGNET LEF Attribute
- Supporting Layout Dual View
- Support for Copying Mixed Signal Routing Constraints from Layout to Abstract View
- Suppressing Messages Issued by Abstract Generator

In the *Running step Abstract* form, use the radio buttons on the left to navigate the options pages for the different steps and bins you are using.

For information about the other abstract generation steps, see the following:

- Pins Step
- Extract Step

Multi-Patterning Technology (ICADVM20.1 Only)

To support Multi-Patterning Technology (MPT), the options on the various tabs of the *Running step Abstract* form now recognize the colors of the original shapes in the layout editor. Therefore, the shapes created in the abstract inherit the colors from the original shapes. For example, two pin shapes created by fracturing a single rectilinear shape will inherit the color of the original rectilinear shape.

Abstract Generator respects color shifts and layers that are present on instances or shapes. For information about color and layer shift functions, see Virtuoso Database Access SKILL Functions for MPT.

© 2020 All Rights Reserved.

For more information about MPT support in Abstract Generator, see Multi-Patterning Technology Support in Abstract Generator (ICADVM20.1 Only).

© 2020 All Rights Reserved.

Adjust Tab

To access the *Adjust* tab, select *Flow – Abstract*. The *Adjust* tab is the first tab in the *Running step Abstract* form.

© 2020 All Rights Reserved.

The options on this tab control how pin shapes are adjusted during connectivity extraction for creating the final required pin shapes. These pin shapes are then fractured into rectangles. The options help you ensure that if a pin shape is reduced in any way, the discarded part of the pin is turned into a blockage, and the blockage is spaced sufficiently away from the reduced pin shape.

The *Adjust* tab provides you options to perform the following tasks:

- Specify the creation of square boundary pins for signal and power nets
- Specify the creation of ring pins for power nets
- Specify the creation of follow ring pins for power nets
- Group the signal nets
- Group the power geometries
- Set the options for CLASS CORE ports
- Create CLASS BUMP ports
- Specify power rail widths, offsets, and shapes
- Assign layers for pin geometry

Creating Boundary Pins

The *Create boundary pins* option exists at two locations on the *Adjust* tab — *Signal Nets* and *Power Nets* groups. By default, this option is off for all bins except the *Block* bin and is disabled and off for *Core* bin.

Square Boundary Pins

The *Create boundary pins* option lets you adjust the pin shapes created during extraction to be square and on the cell boundary. This can be useful when processing blocks where pin geometry extends deep inside the block after extraction. If the *Create boundary pins* option is turned off, the pin will retain the shape generated by the Extract step, which can still be modified using the Assigning Layers for Signal Extraction options.

The method used by Innovus to check width-dependent spacing rules might lead to false violations, so you should restrict the pin geometry to be a square near the boundary of the block by using the *Create boundary pins* option.

© 2020 All Rights Reserved.

Consider the following net inside a block.

With the *Create boundary pins* option turned on, this becomes:

The pins that originally extended inside the block are transformed into squares of the same dimension as the original width of the pin.

Important

If the area of the metal wire from which a boundary pin is cut is less than the minArea constraint defined in the technology file for that layer, a warning message indicating the minimum area rule violation is displayed.

In other words, when verifying the boundary pin size, Abstract Generator does not calculate the metal area of the square boundary pin generated in the abstract. Instead, it calculates the area of the metal wire from which the boundary pin is cut. If the area of the metal wire is less than the minArea constraint, a warning message is generated indicating that the minimum area rule is violated.

Non-Square Boundary Pins

Sometimes, the resultant pin shape might not be square even when *Create boundary pins* is turned on. This can happen when you do not run any extraction but instead use preset pins.

© 2020 All Rights Reserved.

Abstract Generator might need to stretch the pin because it might not be positioned right at the edge of the underlying geometry.

Note: Abstract Generator stretches pins only up to the PR boundary and not beyond it.

Example

Consider the following figure.

The pin does not extend right to the edge of the underlying geometry but the distance ‘d’ is less than the minimum allowed: a separation plus the minimum wire width.

When the *Create boundary pins* option is selected, Abstract Generator first squares off the pin to the right, maintaining the existing left boundary and adjusting the right boundary (the darker hatched area). Then, because distance ‘d’ is too small to accommodate a visible spaced blockage, Abstract Generator stretches the pin back to the boundary (the lighter hatched area).

© 2020 All Rights Reserved.

When the *Create boundary pins* option is not selected, the pin is not squared off first, and the right-most edge is the same as that of the original pin shape. However, Abstract Generator still stretches the pin to the left edge of the underlying geometry as shown in the figure below.

For more information and another example of pin stretching, see Automatic Pin Stretching.

Specifying the Distance of the Boundary Pin from the Boundary

The *Boundary pin max distance to boundary* field in both the *Signal Nets* and *Power Nets* groups on the *Adjust* tab allows you to specify the maximum distance (in microns) of the signal net or power net geometries from the boundary so that they can be considered as boundary pins. The geometries that are within the specified distance from the boundary are squared. By default, the shape(s) closest to the boundary is squared.

This option is available only when the *Create boundary pins* option is selected.

Creating Ring Pins

The *Create ring pins* option is available only if you are processing a block cell. The selection of this option is also mutually exclusive from the *Create boundary pins* option within the shared *Power Nets* section. By default, this option is turned off.

If you select this option, ring pins will be created where any extreme shapes (that are close to the cell boundary) existing in a power net are found to form a ring.

To create ring pins for these power rings, you must ensure that this option, along with the *Extract power nets* option on the *Power* tab in the Extract step, is turned on.

If a block cell is processed with this option selected, Abstract Generator will display a message box informing you whether or not the ring pins were successfully created.

© 2020 All Rights Reserved.

Specifying the Distance of the Ring Pin from the Boundary

The *Ring pin max distance to boundary* field is only available if you are processing a block cell and when the *Create ring pins* option is already selected.

In this field, you can specify (in microns) the maximum distance within which the extracted power net geometry can be from the boundary in order for it to be considered as part of the ring, and therefore, be included in the abstract as pin geometry.

Any `vdd` and `vss` net geometry found further than the specified distance from the boundary is not considered part of the ring geometry but is instead treated as an obstruction.

The default value for this field is `0`, which means that Innovus finds the closest power shape to the boundary for each net, and that becomes the criterion distance used in determining the ring pin geometry for that net.

Creating Follow Ring Pins

The *Follow ring pin* option is available only if you are processing a block cell and when the *Create ring pins* option is already selected. If you select this option, all pin shapes, including the ones that are not part of the ring, are considered as pins in the final abstract.

The pins that are not part of the ring ('follow rings pins') are included in the final abstract based on the value specified in the *Ring pin max distance to boundary* field. When no value is specified, only the pin shapes whose distance from the boundary is the same or less than the distance of the farthest edge of the ring pin shape are included in the abstract. On the other hand, when a value is specified in the *Ring pin max distance to boundary* field, pin shapes are included in the abstract only if their distance from the boundary is within the distance specified in the field.

To prevent any filtering based on the distance, you can specify an arbitrarily large value in the *Ring pin max distance to boundary* field.

Grouping Signal Nets

The *Signal geometry groups* option controls how signal net shapes are grouped into LEF PORTS. This is useful for blocks with numerous signal pins on different layers. Usually, the presence of overlapping shapes on different layers means that the router should connect to only one of the overlapping shapes. This is achieved by grouping the overlapping shapes into the same LEF PORT.

© 2020 All Rights Reserved.

Signal geometry groups is a cyclic field and provides the following options:

single	Creates a single LEF `PORT` for all the signal net shapes
separate	Creates a separate LEF `PORT` for each signal net shape
overlap	Forms LEF `PORTS` from groups of overlapping signal net shapes, that is, signal net shapes that overlap go into the same LEF `PORT`.

The default value for this option is *single* for *Core*, *IO*, and *Corner* bins and *overlap* for *Block* bins.

Tip

Set a value for this option after you have selected the *Create boundary pins* option for signal nets.

Grouping Power Geometries

The *Power geometry groups* option controls how power shapes are grouped into LEF `PORTS`. This is useful for blocks with numerous power pins on different layers.

Usually, the presence of overlapping power shapes on different layers means that the router should connect to only one of the overlapping shapes. This is achieved by grouping the overlapping shapes into the same LEF `PORT`.

Power geometry group is a cyclic field and provides the following options:

single	Creates a single LEF `PORT` for all the power shapes
separate	Creates a separate LEF `PORT` for each power shape
overlap	Forms LEF `PORTS` from groups of overlapping power pins, that is, power shapes that overlap go into the same LEF `PORT`.

The default value for this option is *single* for *Core*, *IO*, and *Corner* bins and *overlap* for *Block* bins.

Tip

Set a value for this option after you have selected the *Create boundary pins* option for power nets.

© 2020 All Rights Reserved.

Examples

```
PIN VDD
      DIRECTION INOUT ;
      USE POWER ;
      PORT
         CLASS CORE ;
      LAYER Metal1 ;
      RECT  22.0000 0.0000 44.0000 11.0000 ;
      END
      PORT
      LAYER Metal1 ;
      RECT  22.0000 0.0000 31.0000 16.0000 ;
      RECT  5.0000 0.0000 20.0000 9.0000 ;
      END
END VDD
```

CLASS CORE Port

Single Port

```
PIN VDD
      DIRECTION INOUT ;
      USE POWER ;
      PORT
         CLASS CORE ;
      LAYER Metal1 ;
      RECT  22.0000 0.0000 44.0000 11.0000 ;
      END
      PORT
      LAYER Metal1 ;
      RECT  22.0000 0.0000 31.0000 16.0000 ;
      END
      PORT
      LAYER Metal1 ;
      RECT  5.0000 0.0000 20.0000 9.0000 ;
      END
END VDD
```

CLASS CORE Port

Two Separate Ports

© 2020 All Rights Reserved.

```
PIN VDD
      DIRECTION INOUT ;
      USE POWER ;
      PORT
         CLASS CORE ;
      LAYER Metal1 ;
      RECT  22.0000 0.0000 44.0000 11.0000 ;
      END
      PORT
      LAYER Metal1 ;
      RECT  22.0000 0.0000 31.0000 16.0000 ;
      END
      PORT
      LAYER Metal1 ;
      RECT  5.0000 0.0000 20.0000 9.0000 ;
      RECT  5.0000 0.0000 23.0000 9.0000 ;
      END
END VDD
```

CLASS CORE Port

Overlap Port

Power Terminal Shape Checks

Abstract Generator also examines the standard cell power pin shapes and sets the shape to either feedthrough or abutment.

Additionally, checks are made at this time to ensure the following:

- The selected standard cells have the same power rail width and the same ground rail width.
- The selected standard cells have the power rails at the same offset from the cell origin so that the power rails can be connected by feedthrough or abutment.
- The rail width is the same at the right and left edges of standard cells.
- The rail width is an even number when converted to LEF database units.

The *Power rail widths, offsets and shape* table provides an option to specify the characteristics of the power and ground rails for the current bin.

The Abstract step does not assign feedthrough or abutment shape properties if there are no power terminal shapes that touch the prboundary. For example, a voltage translator cell could have `vdd` and `vss` net geometry, but may also have an internal `vdd` net that cannot be classified under any shape class.

Note: You can set the shape manually using Viewing and Editing Cell Terminal Properties.

© 2020 All Rights Reserved.

Setting Options for CLASS CORE Ports

The subsequent sections will describe options to specify the following CLASS CORE-specific information to Abstract Generator:

- Direction of the core-facing cell edge
- Power or ground nets to have CLASS CORE ports
- Multiple ports to be defined as CLASS CORE
- CLASS CORE ports to be duplicated in LEF
- Pins to be included in CLASS CORE ports

These options apply only to the cells in the *IO* bin.

Specifying the Direction of the Core-Facing Cell Edge

The *Cell edge facing core* is a text field, which is enabled only when you process a pad cell in the *IO* bin. This option is used to specify the direction in which the `CORE`-facing edge is pointing, *north*, *south*, *east*, *west,* or a combination of any of the four core edges. The default value is *north*. You can specify multiple `CORE`-facing edges.

Abstract Generator uses this value to determine the edge of the pad cell that it should check for locating the port to be classified as `CLASS CORE` in the LEF file.

For example, the IO pad shown in the following figure will be placed to the left of the core. Therefore, the *Cell edge facing core* option for this pad should be set to *east* so that the correct port is identified as CLASS CORE in the LEF file.

Specifying the Power/Ground Nets with CLASS CORE Ports

The *Power/ground net to have CLASS CORE ports* field allows you to specify the nets that should have `CLASS CORE` ports. This can be specified by entering a regular expression in the field. This field is related with the *Cell edge facing core* option and therefore, available only when you process a pad cell in the *IO* bin.

© 2020 All Rights Reserved.

The default entry for this field is `^((V(DD|CC))|(v(dd|cc)))(!)?$`

For more information about completing and understanding regular expressions, see Using Regular Expressions in Abstract Generator.

Allowing the Creation of Multiple CLASS CORE Ports

The *Allow multiple CLASS CORE ports* option is intended for pad cells processed in the *IO* bin. If selected, this option specifies that two ports equidistant from the core-facing edge of a pad are both assigned the `CLASS CORE` statement in LEF. Only geometries with center lines perpendicular to the core-facing edge are considered.

The option is off by default, which means that only the geometry closest to the core-facing edge becomes a `CLASS CORE` port. In the figure below, the geometry that becomes a `CLASS CORE` port is the one on the right (`VDD2`).

When the option is off and there are two power geometries equidistant from the core-facing edge, only the geometry closer to the center line of the pad is defined as a `CLASS CORE` port in LEF. In the figure below, the geometry that becomes a `CLASS CORE` port is the one on the right (`VDD2`).

© 2020 All Rights Reserved.

When you select this option, the behavior depends on the *Power geometry groups* setting, as described below:

- If you select *separate*, the geometries shown in the second picture above appear in two separate `CLASS CORE` ports in LEF.
- If you select *single*, they appear in the same `CLASS CORE` port.
- If you select *overlap*, they appear in two separate `CLASS CORE` ports *unless* they overlap, in which case they will appear in the same `CLASS CORE` port.

Copying CLASS CORE Ports to LEF

The *Copy CLASS CORE ports* option is enabled only when you process a pad cell in the *IO* bin and is switched off by default.

This option duplicates any geometry found to belong to a `CLASS CORE` port on the net (set by using the *Power/ground net to have CLASS CORE ports* option) in a non-`CLASS CORE` port to the exported abstract LEF.

When this option is off, Abstract Generator will find the power geometry group port (*single*, *separate*, *overlap*) whose geometry is closest to the core-facing edge and set it to `CLASS CORE`. Then, when exported to LEF, the port will only be represented once with the `CLASS CORE` property.

Specifying the Pins to be Included in CLASS CORE Ports

The options *Create CLASS CORE ports only if pin meets cell boundary* and *Create CLASS CORE ports if pin meets non-core facing edge* are enabled only when you process a pad cell in the *IO* bin.

Create CLASS CORE ports only if pin meets cell boundary

The option *Create CLASS CORE ports only if pin meets cell boundary* is switched off by default.

When this option is selected, only the pins that either touch or cross the core-facing edge of the boundary become `CLASS CORE` ports in LEF. If the option is not selected, the pin closest to the core-facing edge becomes the `CLASS CORE` port.

© 2020 All Rights Reserved.

Consider the situation as shown in the figure below. In this situation, if the option is selected, then only `pinB` will become a `CLASS CORE` pin.

Example

Consider a situation in which the cell-edge facing core is set to south so the VDD pin on the right of the south edge becomes CLASS CORE in the LEF. Here is how this would look in the LEF file:

```
PIN VDD
      DIRECTION INOUT ;
      USE power ;
      PORT
         CLASS CORE ;
      LAYER Metal1 ;
      RECT  40.280 0.000 42.120 17.680 ;
      END
      PORT
      LAYER Metal2 ;
      RECT  0.400 9.600 43.600 19.600 ;
      RECT  0.400 25.200 43.600 35.600 ;
      RECT  0.400 47.200 43.600 49.600 ;
      RECT  0.400 56.400 43.600 58.800 ;
      LAYER Metal1 ;
      RECT  1.880 0.000 3.720 10.480 ;
      END
END VDD
```

Create CLASS CORE ports if pin meets non-core facing edge

The option *Create CLASS CORE ports if pin meets non-core facing edge* is switched on by default. When this option is selected, pins that touch the non-core facing edge — left

© 2020 All Rights Reserved.

or right (when the core-facing edge is north or south) — become eligible to be included as CLASS CORE ports. The eligible pins become CLASS CORE ports if they satisfy other criteria (settings of other options on the *Adjust* tab).

When the *Create CLASS CORE ports if pin meets non-core facing edge* option is off, pins touching the non-core facing edge are not considered for the CLASS CORE port creation.

Examples:

Consider the following scenarios.

- *Create CLASS CORE ports if pin meets non-core facing edge* = ON

  Create CLASS CORE ports only if pin meets cell boundary = ON

  Assume that both pinA and pinB touch non-core facing edges. With the option *Create CLASS CORE ports if pin meets non-core facing edge* selected, both pinA and pinB are eligible to be considered as CLASS CORE ports

  Since in this situation the *Create CLASS CORE ports only if pin meets cell boundary* option is also ON, only the pin touching the non-core facing edge that also touches the core-facing edge becomes the CLASS CORE port, which in this case is pinA.

- *Create CLASS CORE ports if pin meets non-core facing edge* = ON

© 2020 All Rights Reserved.

Create CLASS CORE ports only if pin meets cell boundary = ON

Core-Facing Edge

In the above figure, only pinA touches the non-core facing edge. However, since both *Create CLASS CORE ports if pin meets non-core facing edge* and *Create CLASS CORE ports only if pin meets cell boundary* options are ON, both pinA and pinB become the CLASS CORE ports.

- *Create CLASS CORE ports if pin meets non-core facing edge* = OFF

  Create CLASS CORE ports only if pin meets cell boundary = ON

Core-Facing Edge

In the above figure, though pinA touches the non-core facing edge but it will not be considered for creating CLASS CORE port because the *Create CLASS CORE ports if pin meets non-core facing edge* option is OFF. In this case, with the *Create CLASS CORE ports only if pin meets cell boundary* option ON, only pinB becomes the CLASS CORE port.

- *Create CLASS CORE ports if pin meets non-core facing edge* = ON

  Create CLASS CORE ports only if pin meets cell boundary = OFF

© 2020 All Rights Reserved.

In this situation, only the pin touching the non-core facing edge that is also the closest to the core-facing edge becomes the CLASS CORE port. In the following figure, the pin that qualifies this criteria is pinB.

Creating CLASS BUMP Ports

The *Create CLASS BUMP ports* option is enabled only when you process a pad cell in the *IO* bin and is switched off by default.

If you select this option, power nets of cell type IO are considered for creating the CLASS BUMP ports. The largest shapes (with the largest Area/Aspect ratio) on the top routing layer are created as bump ports. A CLASS BUMP pin can be used either for power or ground nets.

Note: A power net can have both CLASS CORE and CLASS BUMP ports.

Specifying Power and Ground Rail Characteristics

The *Power rail widths, offsets and shape* table provides an option to specify the characteristics of the power and ground rails for the current bin. The table contains a separate row for each power rail. You can insert or delete rows from the table by selecting the appropriate button, *Add* or *Delete*. Check the box to the left of a row to mark it for deletion.

© 2020
All Rights Reserved.

The *Power rail widths, offsets and shape* option will be unavailable if the *Create ring pins* option is selected. The *Create ring pins* option is available only if you are processing a block.

Caution

If you try to process a block in the Core bin, the Power rail widths, offsets and shape option will lead to a long run time. To avoid this, process blocks in the Block bin.

The fields of the table are described below.

Net

Specify the net name in this column.

Shape

This is a cyclic field and you can select from *feedthru*, *abutment*, *calculate* and *none.*

The table below describes the impact and resultant of selecting one of these values in the *Shape* column.

Net *Shape*	Implication	Default Setting
feedthru	The *Width* and *Offset* values do not need to be set because no calculation is performed. The terminal associated with that net will be given this shape property	Available for *IO*, *Corner*, and *Block* bins.
abutment		
calculate	The *Width* and *Offset* values must be entered for ALL nets because all power rail nets must be calculated.	Available only for cells in the *Core* bin.
none	No calculation will be performed.	Available for *IO*, *Corner*, and *Block* bins.

© 2020 All Rights Reserved.

Note: You cannot set the same power rail net more than once with different *Shape* values associated with it.

If the *Power rail widths, offsets and shape* option is set to *feedthru, abutment* or *none,* Abstract Generator will only use the specified values to set terminal properties. That is, if only one power rail net is entered into the table, that will be the only rail used to set terminal properties.

If Abstract Generator establishes that a rail can be represented by either a feedthru or an abutment rail, it will always choose the abutment shape, as shown below.

```
Abstract: 2001/10/31 14:51:52:

Info:     Power Rail Analysis:

==============================================================================
Rails found
==============================================================================
rail  type    net  layer   width  offset  shape     direction   cells
------------------------------------------------------------------------------
1     power   vdd  met1    2.5    10.87   abutment  horizontal  all
2     power   vdd  met1    5.645  7.725   feedthru  horizontal  all
3     ground  vss  met1    2.8    1.51    abutment  horizontal  all
4     ground  vss  met1    3.965  1.51    feedthru  horizontal  all

Abstract: 2001/10/31 14:51:52:

Info: Using power rails 1 and 3 to set terminal properties.These settings can be
altered using the Adjust “Power rail widths and offsets”  table.
```

Width

Specify the rail width (in μ) in this column.

Offset

Specify the rail position (in μ) in this column.

If the table is left empty, Abstract Generator will try to determine the rail data for the *Core* bin, including calculating the rail and its shape. However, for *Block* and *IO* bins, no calculations are performed if the table is left empty. This is the default situation.

Example Power Rail Scenarios

To establish a better understanding of how this functionality operates, consider the following scenarios:

- Scenario 1: No rail data has been specified

© 2020 All Rights Reserved.

- Scenario 2: No rail data has been specified
- Scenario 3: Values are specified for the net name and rail width and offset
- Scenario 4: No rail data has been specified

Scenario 1: In this scenario, no rail data has been specified and the power rail analyser has attempted to find rails in each of the cells.

At the end of the analysis phase, a table is produced listing the rails found and the cells that contain them. Also listed in the table are those cells where no rails were found, along with any cells that did not have power or ground nets (in this case there are no such cells).

The cells share common rails, so they are used to initialize the power rail table option.

```
Abstract: 2001/10/26 08:18:59:

Info:     Power Rail Analysis:

==============================================================================
Rails found
==============================================================================
rail  type    net  layer   width  offset  shape     direction   cells
------------------------------------------------------------------------------
1     power   vdd  METAL1  6      46.5    abutment  horizontal  all
2     ground  vss  METAL1  6      -1.5    abutment  horizontal  all

Abstract: 2001/10/26 08:18:59:

Info: Using power rails 1 and 2 to set terminal properties. These settings can be
altered using the Adjust “Power rail widths, offsets and shape”  table.
```

Scenario 2: In this scenario, no rail data has been specified by the user in the *Power rail widths, offsets and shape* option table.

Viewing the table below, it can be established that no common rails were found, therefore, the option table was not initialized. This usually occurs when cells have mismatched rails and should be in different bins.

```
Abstract: 2001/10/26 08:55:27:

Info:     Power Rail Analysis:

==============================================================================
Rails found
==============================================================================
rail  type    net  layer   width  offset  shape     direction   cells
------------------------------------------------------------------------------
1     power   vdd  METAL1  4      48.5    abutment  horizontal  sc_a2iaia
                                                                 sc_a2iaia_10
                                                                 sc_a2iaia_11
                                                                 sc_a2iaia_12
                                                                 sc_a2iaia_2
                                                                 sc_a2iaia_3
```


© 2020 All Rights Reserved.

```
                                                                  sc_a2iaia_4
                                                                  sc_a2iaia_5
                                                                  sc_a2iaia_6
                                                                  sc_a2iaia_7
                                                               more ... total of 12

2     power   vdd  METAL1  6      46.5    abutment  horizontal  sc_and8b
                                                                  sc_and8b_10
                                                                  sc_and8b_11
                                                                  sc_and8b_12
                                                                  sc_and8b_2
                                                                  sc_and8b_3
                                                                  sc_and8b_4
                                                                  sc_and8b_5
                                                                  sc_and8b_6
                                                                  sc_and8b_7
                                                               more ... total of 12

3     ground  vss  METAL1  4      -1.5    abutment  horizontal  sc_a2iaia
                                                                  sc_a2iaia_10
                                                                  sc_a2iaia_11
                                                                  sc_a2iaia_12
                                                                  sc_a2iaia_2
                                                                  sc_a2iaia_3
                                                                  sc_a2iaia_4
                                                                  sc_a2iaia_5
                                                                  sc_a2iaia_6
                                                                  sc_a2iaia_7
                                                               more ... total of 12

4     ground  vss  METAL1  6      -1.5    abutment  horizontal  sc_and8b
                                                                  sc_and8b_10
                                                                  sc_and8b_11
                                                                  sc_and8b_12
                                                                  sc_and8b_2
                                                                  sc_and8b_3
                                                                  sc_and8b_4
                                                                  sc_and8b_5
                                                                  sc_and8b_6
                                                                  sc_and8b_7
                                                               more ... total of 12

Abstract: 2001/10/26 08:55:27:

Warning: Cannot find power and ground rails common to all cells in the bin. Specify
using the Adjust “Power rail widths, offsets and shape”  table.
```

Scenario 3: Using the same library used in Scenario 2, the *Power rail widths, offsets and shape* option table is set with values of (`vdd 6 46.5`) and (`vss 6 -1.5`).

This time cells are split into two separate tables, depending on whether they contain the user- specified rails.

```
Abstract: 2001/10/26 10:52:47:

Info:     Power Rail Analysis:
```


© 2020 All Rights Reserved.

```
================================================================================
User specified rails
================================================================================
rail  type    net  layer   width  offset  shape     direction   cells
--------------------------------------------------------------------------------
1     power   vdd  METAL1  6      46.5    abutment  horizontal  sc_and8b
                                                                 sc_and8b_10
                                                                 sc_and8b_11
                                                                 sc_and8b_12
                                                                 sc_and8b_2
                                                                 sc_and8b_3
                                                                 sc_and8b_4
                                                                 sc_and8b_5
                                                                 sc_and8b_6
                                                                 sc_and8b_7
                                                              more ... total of 12

2     ground  vss  METAL1  6      -1.5    abutment  horizontal  sc_and8b
                                                                 sc_and8b_10
                                                                 sc_and8b_11
                                                                 sc_and8b_2
                                                                 sc_and8b_3
                                                                 sc_and8b_4
                                                                 sc_and8b_5
                                                                 sc_and8b_6
                                                                 sc_and8b_7
                                                              more ... total of 12

================================================================================
Rails found in cells not containing user specified rails
================================================================================
rail  type    net  layer   width  offset  shape     direction   cells
--------------------------------------------------------------------------------
1     power   vdd  METAL1  4      48.5    abutment  horizontal  sc_a2iaia
                                                                 sc_a2iaia_10
                                                                 sc_a2iaia_11
                                                                 sc_a2iaia_12
                                                                 sc_a2iaia_2
                                                                 sc_a2iaia_3
                                                                 sc_a2iaia_4
                                                                 sc_a2iaia_5
                                                                 sc_a2iaia_6
                                                                 sc_a2iaia_7
                                                              more ... total of 12

2     ground  vss  METAL1  4      -1.5    abutment  horizontal  sc_a2iaia
                                                                 sc_a2iaia_10
                                                                 sc_a2iaia_11
                                                                 sc_a2iaia_12
                                                                 sc_a2iaia_2
                                                                 sc_a2iaia_3
```


© 2020 All Rights Reserved.

```
                                                                    sc_a2iaia_4
                                                                    sc_a2iaia_5
                                                                    sc_a2iaia_6
                                                                    sc_a2iaia_7
                                                                more ... total of 12
```

Scenario 4: In this scenario, the user has not specified any rail data.

Because common rails have not been found in all cells in the bin, the *Power rail widths, offsets and shape* option table is not initialized. Note, however, that cells containing no rails at all were found, as were cells that did not contain power and ground nets.

The scenario where a cell has no rail identified can arise because the net geometry does not extend far enough out to the edge of the cell. Abstract Generator looks for rail geometry in a corridor within 10% of the dimension of the place-and-route boundary.

Cells that do not contain at least one power *and* ground net pair are also flagged. This might occurs if an incorrect text to pin map has been used during the Pins step, or if text labels do not overlap metal geometry.

```
Abstract: 2001/10/26 11:55:41:

Info:     Power Rail Analysis:

==============================================================================
Rails found
==============================================================================
rail  type    net  layer   width  offset  shape     direction   cells
------------------------------------------------------------------------------
1     power   vdd  METAL1  6      46.5    abutment  horizontal  sc_a2iaia
                                                                 sc_a2iaia_10
                                                                 sc_a2iaia_11
                                                                 sc_a2iaia_12
                                                                 sc_a2iaia_13
                                                                 sc_a2iaia_14
                                                                 sc_a2iaia_15
```


© 2020 All Rights Reserved.

```
                                                                  sc_a2iaia_16
                                                                  sc_a2iaia_17
                                                                  sc_a2iaia_18
                                                               more ... total of 25

2     ground  vss  METAL1  6      -1.5    abutment  horizontal  sc_a2iaia
                                                                  sc_a2iaia_10
                                                                  sc_a2iaia_11
                                                                  sc_a2iaia_12
                                                                  sc_a2iaia_13
                                                                  sc_a2iaia_14
                                                                  sc_a2iaia_15
                                                                  sc_a2iaia_16
                                                                  sc_a2iaia_17
                                                                  sc_a2iaia_18
                                                               more ... total of 25


No rails found in any of the following cells:
---------------------------------------------
sc_and2a sc_and2a_10 sc_and2a_11 sc_and2a_12
sc_and2a_13 sc_and2a_14 sc_and2a_15 sc_and2a_16
sc_and2a_17 sc_and2a_18 sc_and2a_19 sc_and2a_2
sc_and2a_20 sc_and2a_21 sc_and2a_22 sc_and2a_23
sc_and2a_24 sc_and2a_25 sc_and2a_3 sc_and2a_4
sc_and2a_5 sc_and2a_6 sc_and2a_7 sc_and2a_8 sc_and2a_9


None of the following cells have power and ground nets:
-------------------------------------------------------
sc_syncdflop sc_syncdflop_10 sc_syncdflop_11 sc_syncdflop_12
sc_syncdflop_13 sc_syncdflop_14 sc_syncdflop_15 sc_syncdflop_16
sc_syncdflop_17 sc_syncdflop_18 sc_syncdflop_19 sc_syncdflop_2
sc_syncdflop_20 sc_syncdflop_21 sc_syncdflop_22sc_syncdflop_23
sc_syncdflop_24 sc_syncdflop_25 sc_syncdflop_3 sc_syncdflop_4
sc_syncdflop_5 sc_syncdflop_6 sc_syncdflop_7 sc_syncdflop_8
sc_syncdflop_9
```

Generating Abstracts for Routed Blocks with Power Rings

There might be situations in which the cell data imported comes from a DEF file. This is usually a routed block for which you might want to create an abstract to use at a higher level of a design.

Abstract Generator can generate abstracts from routed blocks. This includes the ability to create abstracts for blocks with ring power pins, for blocks with cover blockages inside the power ring, and for blocks with detailed obstructions for signal nets crossing the power rings.

Power Pins

Consider the following points when deciding which approach to use when dealing with power ring pins:

© 2020 All Rights Reserved.

1. Before you decide whether or not power extraction is necessary.

   Power extraction is not required if, after running the Pins step, the power pins contain at least the power pin geometry. They will contain power pin geometry if the power ring shapes were defined as physical pins in the imported DEF file. This, however, is unlikely, and in the case where no physical pins were defined in the DEF file, the Pins step will create pins from all of the power routing in the block (including the power ring).

   If extraction is required, perform a limited power extraction by limiting the hierarchy depth. This will result in the extraction of the power ring plus some other top level shapes. As a recommendation, set extraction depth to `1` so that Abstract Generator finds the ring pins plus the via geometry. Go to step 2.

   If there is no need to extract, leave power extraction turned off and consider the Abstract step requirements in step 2.

2. In the Abstract step, ensure that the *Create ring pins* option is turned on. The algorithm to determine if a power pin should be preserved as a ring pin is based on *distance to boundary*. The closest power pin to the boundary is selected along with any others that are equidistant. Therefore, if a power ring is equidistant to the boundary on all sides, then everything should be fine.

   Otherwise, you will have to manually control the settings in the *Ring pin max distance to boundary* field, setting it to be equal to the maximum distance from the boundary within which you want any power shapes to be considered as ring pins.

   Note: You must set this based on the distance to the *inner ring* (VDD or VSS, whichever is inner).

Signal Pins

If the signal pins read in from the DEF file are sufficient, there is no need to perform connectivity extraction, and no requirement to have boundary pins turned on. Therefore, turn the *Create boundary pins* option off since it is on by default in the *Block* bin.

Alternatively, if your signal pins are insufficient, you would run connectivity extraction and leave the *Create boundary pins* option turned on.

Obstructions

To form the necessary obstructions with the region inside the power ring as a solid obstruction and with the signals that cross the ring into obstructions in that region, set the *Blockage* option to *Cover* in the *Layer Assignment for Blockages* table and specify the cover blockage distance to *Boundary* value.

© 2020 All Rights Reserved.

Blockage Tab

To access the *Blockage* tab, select *Flow – Abstract* and then click the *Blockage* tab in the *Running step Abstract* form.

The options on this tab control the final blockage geometry in the abstract view.

Note: Any blockage geometry already existing in the view is preserved and added to the final calculated blockages.

© 2020
All Rights Reserved.

Assigning Layers for Blockages

The *Layer Assignment for Blockages* table is used to specify how blockages are to be modeled in the abstract. This table consists of the following columns:

- Layer
- Geometry Specification
- Blockage
- Pin Cutout
- Max Space
- Corridor Cut
- Cut Same
- Cut Below
- Shrink Dist
- Shrink Track
- Boundary

Layer

Specify the layers on which you want to create blockages. You can do one of the following:

- Select a layer from the drop-down list available from the technology file.

  After you have selected a layer from the drop-down list, it is removed from the list for the next layer row.

- Type the layer name directly in the *Layer* column.

  Specify each layer only once. If you enter an invalid layer, Abstract Generator will issue a warning.

If you do not create a table entry for a particular layer, Abstract Generator will not create any blockages on that layer in the final abstract.

Geometry Specification

Enter the Geometry Specification for the layers on which you want the blockage to be implemented.

© 2020 All Rights Reserved.

Blockage

Choose the type of blockage you want to create on the specified layer: *Cover*, *Detailed*, or *Shrink*. The default is *Detailed* for the *Core* bin and *Cover* for the *Block* bin.

Cover	A *Cover* blockage is a blockage that covers an entire block for each metal layer used by the block. This type of blockage is typically generated for blocks or IO cells where you want to prevent over-the-cell routing to improve performance or to avoid electrical effects between tracks. Cover blockages follow the overlap layer, if present; else they follow the PR boundary. The cover blockage model does not generate any blockages outside PR boundary. Only exceptions to this are the shapes that are affected by pin cutout regions, where these shapes are considered to be part of the layout.
Shrink	A *Shrink* blockage is a blockage that fills in the smaller, less useful free spaces in a block and leaves out the larger spaces. Based on the specified Shrink Dist or Shrink Track, Abstract Generator combines blockages within that distance of each other, filling the space between them with more blockage. This results in a small number of large blockages separated by spaces that a router can use. This allows over-the-cell routing without modeling each obstruction individually. The *Shrink* blockage generation slows down the overall process of abstract generation but ultimately increases the speed of routing significantly, which is better than when *Detailed* blockages are routed.
Detailed	A *Detailed* blockage is a blockage that covers geometries that are not part of pins on the routing layers, thereby allowing maximum routing within the block. This type of blockage is used for standard cells and possibly for higher-level routing layers in blocks. Do not create detailed blockages on contact or via layers for large blocks. Creating detailed blockages can adversely affect the performance of Abstract Generator because of the number of individual shapes that need to be processed. Consider creating *Shrink* type of blockage by setting a distance to merge most of the obstructions. You can speed up the detailed blockage creation in some scenarios. For more information about these scenarios, see Improving the Performance of Detailed Blockage Generation.

© 2020 All Rights Reserved.

Improving the Performance of Detailed Blockage Generation

If you are creating *Detailed* blockages for layouts that have big blockages at the top level, you can gain considerable performance benefits by following the use model described here.

This use model is implemented only if the geometry specification (see the criteria below) for a blockage layer is such that the top-level shapes that satisfy the geometry specification are not excluded when the same geometry specification is applied at all the levels. In such a situation, the shapes are processed in the following way:

- At the top level, shapes satisfying the geometry specification and that are bigger than 10 percent of the total cell area are considered as “big blockages”.
- At the lower levels, shapes on the boundary purpose and that are bigger than 10 percent of the total cell area are considered as “big blockages”.

After the “big blockages” are identified, they are processed in the same way as *Cover* blockages. This use model greatly reduces the complexities of processing and merging shapes and also reduces the number of shapes that are ultimately processed for blockage creation.

Tip

Use the `BlockageLargeShape` option to enable Abstract Generator to look for big blockages in the current design. The `BlockageLargeShape` option is always used in association with the `BlockageLargeShapePct` option. The size of the identified blockages is increased by the percentage specified by the `BlockageLargeShapePct` option, and a cover blockage is created.

Use the `BlockageLargePurposeList` option to specify the purpose of the identified big blockages. For more information, see the options for the `absAbstract` SKILL function.

Use the following criteria for writing a valid geometry specification:

- Geometry specifications with a single purpose:
  - The boundary purpose should be specified explicitly. For example,

    ```
    M1 boundary
    ```

  - If any purpose other than the boundary purpose is specified, there should be no value specified in the *Ignored Purpose(s)* field (*File – General Options – General* tab). For example,

    ```
    M1 net
    M1 drawing
    ```


© 2020 All Rights Reserved.

- Geometry specifications with more than one purpose:
  - The boundary purpose should be specified along with the other purpose(s) by using only the OR operator. For example,

    ```
    M1 boundary or M1 drawing
    ```

  - If any combination of LPPs excluding the boundary purpose is specified, there should be no value specified in the *Ignored Purpose(s)* field (*File – General options – General* tab). For example,

    ```
    M1 net or M1 drawing
    ```

Pin Cutout

Enables Abstract Generator to cut the blockage around pins for allowing a router to access the pins. For more information about how this option affects the behavior of the three blockage types, see Impact of Cutout and Spacing Options on the Blockage Types.

Max Space

Enables Abstract Generator to create a boundary blockage around a block such that the top-level routes are maximum possible distance away from the block. The boundary blockage is interspersed between pins along the block boundary. You can use this option when you want to do routing for cells even when their detailed layout is still not complete but the pin positions and boundary have been finalized.

This option works only for *Cover* blockages and is not applicable for *Detailed* and *Shrink* blockages. For more information about how this option affects the behavior of the three blockage types, see Impact of Cutout and Spacing Options on the Blockage Types.

The *Max Space* option is effective only when the Pin Cutout option is selected. Further, when the *Max Space* option is selected along with the *Pin Cutout* option, the Corridor Cut option is ineffective.

Controlling Spacing Value During Geometry Add Backs

When Abstract Generator creates a cover blockage with pin cutout, it automatically adds back geometries that are located near the boundaries. Geometries that are located between the metal_max_spacing value and metal_min_spacing value are added back. Depending on your requirement, you can specify the distance within which the geometries need to be added back. For example, for some designs, a large number of geometries may be located within this distance. In such cases, instead of adding back all the geometries, you can reduce the spacing value. Use the `absSetBinOption( "Core" "AbstractBlockageUserDefinedDistanceForAddBack" "")` option to control the spacing values.

© 2020 All Rights Reserved.

Example

```
absSetBinOption( "Core"
"AbstractBlockageUserDefinedDistanceForAddBack" "(M1 0.5) (M2
0.6)")
```

In the above example, for M1 cover blockage, add back geometries will be searched within distance of 0.5 and for M2 cover blockage, add back geometries will be searched within the distance of 0.6.

In the cover blockage model, all shapes of the specified layer that are close to or are partially overlapping with the PR boundary are retained, and created as addbacks. Shapes of the specified layer that are deep inside the PR boundary or are completely outside the PR boundary are lost.

In the detailed blockage model, all shapes of the specified layer that are inside, outside, or overlapping the PR boundary are created as blockages.

Corridor Cut

Enables Abstract Generator to create corridor paths that can allow routers to access internal pins. This option, when selected, first determines the existence of straight line paths that can route `minWidth` wires from the internal pins to the block boundary and then creates the corridor cuts. Also, Abstract Generator reads the direction of the pin and checks the possibility of creating corridor cut in a direction. If the pin direction matches with the nearest edge then the corridor is cut in that direction. However, if the pin direction does not match with the direction of the nearest edge then the corridor will not be cut and an error message will be displayed.

For more information about how this option affects the behavior of the three blockage types, see Impact of Cutout and Spacing Options on the Blockage Types.

You can use this option only when the Pin Cutout option is selected.

Cut Same

Specifies the distance to cut around pins on the same layer. If set as `0`, the blockage completely covers the design, including the pin shape, and there is no pin cut out.

By default, Abstract Generator cuts out enough space to enable a via to be dropped from the layer above anywhere onto the pin without violation. This setting overrides the default size of the cut-out made around pins. The default is still used for any unlisted blockage layers.

See also Automatic Pin Stretching.

© 2020 All Rights Reserved.

Cut Below

Specifies the distance to cut around pins on the layer below.

By default, Abstract Generator cuts a window from a blockage layer that overlaps pins on the layer below, leaving enough space to enable a via to be dropped from the blockage layer anywhere onto the pin without violation. For example, if you are creating cover blockages for metal 4, then pin cutouts are created on metal 4 blockages for pins in both, metal 4 and metal 3.

By default, the *Cut Below* field does not have any value. However, you can specify any value in this field. If you specify the value 0, then pin cutouts are created only on the current layer. In the above example, pin cutouts will only be created on metal 4.

See also Automatic Pin Stretching.

Shrink Dist

Specifies a minimum distance (in microns) between blockages. Blockages separated by a distance less than or equal to the specified distance are merged, irrespective of whether Pin Cutout is ON or OFF. Values entered in this field are considered only if you select to generate *Shrink* blockages. For more information about the *Shrink* blockage, see Blockage.

Shrink Track

Specifies the minimum number of tracks between blockages. Blockages separated by a distance less than or equal to the specified number of tracks are merged. Values entered in this field are considered only if you select to create *Shrink* blockage. For more information about *Shrink* blockage, see Blockage.

The distance, `D`, required to accommodate a given number of tracks, `n`, is calculated (in microns) using the following formula:

```
D = (n * minTrackWidth) + ((n + 1) * minSpacing)
```

Important

If you specify both *Shrink Track* and *Shrink Dist*, the values are added and blockages separated by a distance less than or equal to the total distance are merged. If you do not supply a distance, Abstract Generator uses a default value of `2` tracks.

If values are entered for layers for which *Shrink* blockage is not being generated, the values are ignored.

© 2020 All Rights Reserved.

Boundary

Specifies the minimum distance between a *Cover* blockage and the cell boundary. The default value of this option is `0`. This option is valid only for the *Cover* blockage. For more information about this blockage type, see Blockage.

The Add, Edit, and Delete buttons and the G-Spec Builder Form in conjunction with the Operators are used to work with the *Layer Assignment for Blockages* table. When you add a new layer row to the table, the type of blockage is set as *Detailed* for the *Core* bin and *Cover* for all other bins by default.

© 2020 All Rights Reserved.

Impact of Cutout and Spacing Options on the Blockage Types

This section describes how each of the three types of blockages behave with the Pin Cutout, Max Space, and Corridor Cut options.

Cover Blockage

A *Cover* blockage covers the entire block. The default *Cover* blockage does not have cutouts either around pins or at the block boundary and is tagged with the spacing attribute `+SPACING` set to `maxSpacing`. Therefore, the top-level routes stay `maxSpacing` away from the block. The *Cover* blockage in this state offers the least cross coupling between the routes at the top level and those within the block. The top-level routes can access pins at the boundary and the pins within the block can be accessed from the top in a perpendicular direction.

For a default *Cover* blockage generation, neither of *Pin Cutout*, *Corridor Cut*, and *Max Space* option is selected.

- *Pin Cutout* is OFF

  Abstract Generator will not create cutouts around pins. A large blockage is added that will cover the entire block.

- *Pin Cutout* is ON

  Abstract Generator creates pin cutouts at `minSpacing` distance away from the pins. However, if the layer-purpose-based and voltage dependent rules are present in the technology file along with layer based spacing rules and option `RespectPurposeAndVoltageRules`, to conform to these rules, is set to TRUE, then Abstract Generator will read the layer-purpose rules, voltage-based rules, and the layer rules. A blockage around the pin is created with the maximum value of the `minSpacing` available based on the layer-purpose-based and voltage dependent rules. The use of the *Pin Cutout* option is recommended for generating a *Cover* blockage to obtain a densely packaged block.

- *Pin Cutout* and *Corridor Cut* are ON

  Abstract Generator creates corridor cuts for internal pins in addition to creating the pin cutouts. The creation of corridor cuts is subject to the existence of straight-line paths from internal pins to the block boundary (for a `minWidth` wire). The *Pin Cutout* option should be selected for selecting the *Corridor Cut* option.

- *Pin Cutout* and *Max Space* are ON

  In addition to the pin cutouts, Abstract Generator creates a blockage at the block boundary that keeps the top-level routes at `maxSpacing` away from the block. This

© 2020 All Rights Reserved.

further reduces the cross coupling between internal and top-level routes. The *Max Space* option is functional only when the *Pin Cutout* option is selected. The *Corridor Cut* option is ineffective when both these options are selected.

Shrink Blockage

A *Shrink* blockage covers the small, less useful areas in a block. The default *Shrink* blockage does not have cutouts either around pins or at the block boundary.

In the case of *Shrink* blockages, no blockage is added around the block at the boundary. Therefore, the *Max Space* option is ineffective in the case of *Shrink* blockages even if selected.

- *Pin Cutout* is ON

  Abstract Generator creates pin cutouts at `minSpacing`.

- *Pin Cutout* and *Corridor Cut* are ON

  Abstract Generator creates corridor cuts for internal pins. The creation of corridor cuts is subject to the existence of paths from internal pins to the block boundary (for a `minWidth` wire). The *Pin Cutout* option should be selected for the *Corridor Cut* option to be effective.

Detailed Blockage

A *Detailed* blockage covers geometries that are not part of pins on the routing layers. The default *Detailed* blockage does not have cutouts either around pins or at the block boundary.

In the case of *Detailed* blockages, no blockage is added around the block at the boundary and no corridors are cut around pins. Therefore, both *Max Space* and *Corridor Cut* options are ineffective in the case of *Detailed* blockages even if selected.

- *Pin Cutout* is OFF

  Abstract Generator does not create cutouts around pins.

- *Pin Cutout* is ON

  Abstract Generator first creates pin cutouts on the *Detailed* blockage. These pin cutouts are computed using the appropriate spacing rules, which in turn, are based on the respective pin's width. Next, Abstract Generator determines instances of pin cutouts in which the cutout width is greater than the `minSpacing` for the layer. For these instances, Abstract Generator adds small blockages between the `minSpacing` and the pin cutouts.

© 2020 All Rights Reserved.

Finally, the small added blockages are tagged with the spacing attribute `+SPACING` set to `minSpacing`.

Note: The additional blockages are added to prevent top-level routes from entering the pin cutout area.

The following table summarizes the effect of selecting various combinations of these options for the three types of blockages. The red button indicates that the option is selected and the white button indicates that the option is not selected. The symbol 'X' for an option indicates the ineffectiveness of the option in that particular scenario even if selected.

Blockage Type	Options in the Table on the Blockage Tab			Resultant Blockage
	Pin Cutout	Max Space	Corridor Cut	
Cover	❑	X	X	Default cover blockage: - No pin cutouts - No corridor cuts - No blockage at the boundary
	■	❑	❑	Cover blockage with pin cutouts. No corridor cuts.
	■	❑	■	Cover blockage with pin cutouts and corridor cuts for internal pins.
	■	■	X	Cover blockage with pin cutouts and blockage at boundary to keep routers `maxSpacing` away. No corridors cuts.
Shrink	❑	X	X	Default shrink blockage: - No pin cutouts - No corridor cuts - No blockage at the boundary
	■	X	❑	Shrink blockage with pin cutouts. No corridor cuts.
	■	X	■	Shrink blockage with pin cutouts and corridor cuts for internal pins.

© 2020 All Rights Reserved.

Blockage Type	Options in the Table on the Blockage Tab			Resultant Blockage
	Pin Cutout	Max Space	Corridor Cut	
Detailed	☐	X	X	Default detailed blockage: - No pin cutouts - No corridor cuts - No blockage at the boundary
	■	X	X	Detailed blockage with pin cutouts. No corridor cuts.

■ Selected
☐ Not Selected
X Not Applicable

© 2020 All Rights Reserved.

Specifying the Size of the Window to be Cut Around a Pin

You can use the *Cut window around pins large enough to drop via* option to control the size of the window cut into a blockage around a pin.

This option is selected by default. When this option is selected, Abstract Generator cuts a window large enough for a via to be dropped to that pin. If you do not select this option, Abstract Generator cuts a window around the pin based purely on the range-based spacing rule for the pin width.

Alternatively, use the `BlockageCutVia` option.

Important

Abstract Generator uses the `minSpacing` value for the minimum spacing for both metal and via layers. It does not use the `viaSpacing` rule. If you have `viaSpacing` tables, you need to ensure that the spacing for zero adjacent cuts is defined in the `minSpacing` rule.

This option applies only to the layer on which the pin and blockage exist. However, for a given blockage layer, Abstract Generator additionally cuts around pins on the layer below, again leaving space for a via to be dropped without violation. This option has no effect on the size of the cut-out around pins on the layer below.

Tip

If you want the cut-outs on all layers to be based only on the pin width, set the *Cut Below* value to `0` for the given blockage layer in the *Layer Assignment for Blockages* table.

Specifying the Effective Width of Blockages

Use the `AbstractBlockageUserDefinedWidth` option to set the effective width attribute for layer blockages. Spacing and effective width attributes cannot be set together. Effective width can be set only if the spacing value is set to zero. For information about how to set this option, see `AbstractBlockageUserDefinedWidth`.

Supporting Pin Cutouts on Metal Layers Above the Pin Layer (for Stacked Vias)

When the Pin Cutout option is enabled, Abstract Generator creates cutouts on the layer that contains pins. To create pin cutouts on blockage layers above the layers that contains pins,

© 2020 All Rights Reserved.

you can use the `AbstractBlockageCutForAboveLayers` option available in the Abstract step of Abstract Generator. Specifying the `AbstractBlockageCutForAboveLayers` option enables the router to connect to pins by using stacked vias. This option is not available in the GUI.

Note: The `AbstractBlockageCutForAboveLayers` takes precedence over all other cut distance related options such as `AbstractBlockageDownPinCutWindow` and `BlockageCutVia`.

With the `AbstractBlockageCutForAboveLayers` option, you specify a valid metal layer that contains pins and the size of the pin cutouts to be created in the blockage layers above this layer. Specifying the size of the pin cutout is al. If the pin cutout size is not specified, the size of the pin cutout is determined by the minspacing of the blockage layer. The following example illustrates this:

```
absSetBinOption("Block" "AbstractBlockageCutForAboveLayers" "(Metal2 0.1) (Metal4
0.2)")
```

Here, for pins on `Metal2` layer, the blockages on `Metal3`, `Metal4`,...,`MetalX` are cut with a distance of `0.1u;` for pins on `Metal4` layer, the blockages on `Metal5`, `Metal6`,...,`MetalX` are cut with a distance of 0.`2u`.

The following is another example that specifies pin cutouts:

```
absSetBinOption("Block" "AbstractBlockageCutForAboveLayers" "Metal2 (Metal4 0.2)")
```

Here, for pins on `Metal4` layer, the blockages on `Metal5`, `Metal6`,...,`MetalX` are cut with a distance of `0.2u`. However, for pins on `Metal2` layer, the blockages are cut with a minimum spacing specified for the layer on which the blockage is created. This is because the pin cutout value is not specified for `Metal2` layer. Therefore, the blockages on `Metal3` are cut with `Metal3` minSpacing and blockages on `Metal4` are cut with `Metal4` minSpacing, and so on.

The `AbstractBlockageCutForAboveLayers` option works only for those layers for which Pin Cutout is enabled. In the example below, pin cutout is enabled only for `Metal2` to `Metal5` layers. Therefore, blockages will be cut only for these layers.

```
absSetBinOption("Block" "AbstractBlockageCutAroundPin" "Metal2 Metal3 Metal4
Metal5")
```

Creating Routing Channels when Generating Cover Obstructions

Abstract generator enables you to create routing channels through cover blockages. Consider layer `Metal1`. When generating a cover obstruction on this layer, you can choose to cut a routing channel through this layer. The routing channel will take the shape of the layer-purpose pair, which you specify as a parameter. In this example, the routing channel will be in accordance with (y9, drawing) layer-purpose pair.

© 2020 All Rights Reserved.

Note: This feature is not available for shrink and detailed blockages.

To create a routing channel when generating a cover blockage, specify the parameters in the *Routing channel for cover blockage* field.

Here, *Metal1* refers to layer through which a routing obstruction needs to be cut. *(y9 drawing)* is the layer-purpose pair that indicates the shape of the routing channel.

ally, you can use the `AbstractBlockageRoutingChannelOnLayers` option to create a routing channel when generating a cover blockage.

Example

```
absSetBinOption("Block" "AbstractBlockageRoutingChannelOnLayers" "(Metal1 (y9
drawing)) (Metal2 (y9 drawing))")
```

Caution

Layers `(y [0-9] drawing)` *are system-defined layers. So, the results may not be accurate when these layers are used for creating routing channels.*

Automatic Pin Stretching

Note: The following section is only for information. There is no option to enable “automatic pin stretching”.

© 2020 All Rights Reserved.

Abstract Generator sometimes requires to make the pin geometry larger than the settings you have specified. This feature is known as *automatic pin stretching* and is used in circumstances where cutting a window around a pin would remove some geometry (obstruction) that touches the pin, potentially letting the router route too close to the cut-out geometry. See the example below.

Important

Automatic pin stretching searches for geometry only to the *Maximum depth* of hierarchy as specified in the Extract step's *Signal* or *Power* tab. The default depth is 32.

Example

This is an example of a situation in which pin stretching occurs. The example also illustrates the reasons why pin stretching occurs.

The following figure shows the wire geometry W touching the Pin P. When a window is cut out around Pin P, the wire geometry that touches the pin at the right edge, W2, is totally removed in the abstract. W1 part of the wire geometry becomes an obstruction. The W2 wire geometry cannot also be converted into an obstruction because it covers the area in which a window has already been cut out around Pin P.

The figure below shows both the outcomes when Pin P is either stretched (**B**) or not stretched (**A**) towards the right side. In **A**, when the pin is not stretched, the removed wire geometry is not represented in the abstract. So it is possible for the router to route a wire within the minimum separation distance to the right side of Pin P. This can lead to spacing violation with the W2 portion of the original wire when the abstract is replaced with the pin and wire geometries in the layout.

© 2020 All Rights Reserved.

Therefore, the W2 wire geometry must be represented in the abstract somehow. In this situation, the pin is automatically stretched to the right to cover the portion of the wire that cannot be turned into an obstruction (W2). Since the resultant stretched pin includes the wire geometry, the router will now space away a new wire away appropriately from the W2 wire geometry. This is shown in **B** in the figure above.

There is no need to extend Pin P vertically above because even when the window is cut out around the pin, there is still a piece of the wire (W1) that is converted into an obstruction above Pin P. Therefore, unless the cut out around the pin is large, the space between Pin P and the new obstruction will not be large enough for the router to route unrelated wiring through.

By default, pin stretching occurs automatically whenever a window cut out around a pin removes some geometry touching the pin. If you want, you can disable pin stretching by setting the *Cut Same* value to 0 and deselecting the *Cut window around pins large enough to drop via* option.

For more information and another example of automatic pin stretching, see Non-Square Boundary Pins.

© 2020 All Rights Reserved.

Density Tab

You can generate metal density information for abstracts in LEF 5.8. The density value in LEF helps in verifying metal density fabrication rules early in the design cycle without having to replace the abstracts with the layouts. This information in LEF can reduce the memory requirements and increase the speed of metal fill tools and aid these tools to analyze and fix the designs using abstracts.

To access the *Density* tab, select *Flow – Abstract*, then click the *Density* tab in the *Running step Abstract* form.

The options on this tab provide specifications to Abstract Generator for calculating metal density (in percentage) for macros.

© 2020
All Rights Reserved.

You select the *Calculate metal density* option to specify that metal density information is to be generated for the selected cell(s). If the *Calculate metal density* option is selected, Abstract Generator generates the fill blockages that are specified in the DEF file. During LEF generation, these fill blockages are exported out as density.

By default, the *Calculate metal density* option on this tab is enabled. The other options become active only when the *Calculate metal density* option is enabled.

Next, you specify the details to enable Abstract Generator in determining the metal layers and metal geometries for which density calculation needs to be performed.

Specifying the Metal Layers for Density Calculation

For metal density calculation, you can either use the already specified metal layers and their geometry specifications or specify them afresh on this tab. These two alternatives are elaborated below:

- You can use the same metal layers and their corresponding geometry specifications as you would have specified for signal nets, power nets, or antenna data extraction in the Extract step. To do this, select one of the following options on the *Density* tab:
  - *Use layer assignment for signal extraction*
  - *Use layer assignment for antenna extraction*
  - *Use layer assignment for power extraction*

  These options are mutually exclusive and selecting one disables the other two options. Only metal layers specified in the selected table will be considered for metal density calculation.

  Along with selecting one of these options, you can ally specify the *Width* and *Height* of the density window in the *Layer Assignment for Metal Density Regions* table. For more information, see Specifying the Dimensions of the Density Window.

- Alternatively, you can specify a different set of geometry specifications for metal layers by using the *Layer Assignment for Metal Density Regions* table. To use the geometry specifications from this table, ensure that none of the three layer assignment options is selected. You can turn off these options individually.

  The table contains the following columns:

  - The *Layer* column is pre-populated with the metal layers from the technology file. You can remove or edit the existing layers or add any removed layers. Density data will not be calculated for metal layers that are not specified in the table.

© 2020 All Rights Reserved.

Note: If you use the *Layer Assignment for Metal Density Regions* table and you have specified a non-metal layer in the table, Abstract Generator will issue a warning message during density calculation.

- By default, the *Geometry Specification* column contains the layer names corresponding to each metal layer name in the *Layer* column. You can modify these values as required.
- The *Width* and *Height* columns are described in the next section, Specifying the Dimensions of the Density Window.

Important

If you select to use a layer assignment table for density calculations from a different tab, you can still use the *Width* and *Height* columns of the *Layer Assignment for Metal Density Regions* table on the *Density* tab. In this case, though the geometry specifications in the *Layer Assignment for Metal Density Regions* table will not be considered for metal density calculation but you need to ensure that a valid geometry specification exists for every layer specified in this table.

Specifying the Dimensions of the Density Window

You can specify the width and height of the density window either for all the layers or separately for each layer.

To specify the width and height on a per layer basis, use the *Width* and *Height* columns of the *Layer Assignment for Metal Density Regions* table. You can use the *Width* and *Height* columns of this table when you either select one of the layer assignment options or use the *Layer Assignment for Metal Density Regions* table for density calculation.

To specify the same width and height value for all the layers, use the following fields on the *Density* tab:

- *Default density window width*
- *Default density window height*

The default value for both the fields is `20`, which is derived by calculating 10% of a typical `130nm` window and 20% of a typical `90nm` density check window size. You can modify this value to get Abstract Generator to override the default window size without having to change the *Width* and *Height* values for every layer.

Note: If you do not populate the *Width* or the *Height* column, Abstract Generator picks the density window dimension from the *Default density window width* and *Default density window height* fields.

© 2020 All Rights Reserved.

Some density windows in the final abstract may be smaller in dimension than the option values specified. This happens around the edges of the PR boundary on various layers. This is illustrated in the following scenarios:

Scenario I: Rectangular PR Boundary

Starting from the lower left corner of the bounding box of the PR boundary, the abstract is divided into a rectangular grid with window size as specified in the GUI. The rectangles intersecting the PR boundary are clipped at the right and top sides. Therefore, the size of the density windows at the right and top sides will be smaller than the size specified for the density window in the GUI. This is shown in the figure below.

Scenario II: Rectilinear PR Boundary

The density windows are clipped wherever they intersect the PR boundary. The resulting polygons are fractured into smaller rectangles and density data is generated for each of these independently.

© 2020
All Rights Reserved.

Storage Location of Metal Density Information

On OpenAccess, the density rectangles are saved as an `oaLayerBlockage` object with type `oacFillBlockageType`. The bBox of the `oaLayerBlockage` object is reported as the metal density rectangle in LEF by the export tool.

© 2020 All Rights Reserved.

Fracture Tab

To access the *Fracture* tab, select *Flow – Abstract*, then click the *Fracture* tab in the *Running step Abstract* form.

The options on this tab let you control whether or not pins and blockages are fractured depending on the PR system you are using. By using the options on the *Fracture* tab, you can also control the modeling of 45-degree geometry. These options are available to all bins.

Enabling Fracturing Options

You can specify whether you want to fracture pins or blockages by using the options in the *Fracture* section - *Fracture pins* and *Fracture blockages*. By default, both these options are disabled.

- *Fracture pins*: This option controls whether Abstract Generator creates each pin shape as a set of maximum rectangles or as a polygon. This lets you model 45-degree pins if your design requires it. This option is available for all bins.

  When this option is not selected, which is the default, Abstract Generator creates each blockage shape as a polygon. This setting is recommended if the abstracts are to be routed using the IC Shape-Based Router.

© 2020 All Rights Reserved.

If you select this option, Abstract Generator creates each pin shape as a set of maximum rectangles.

You can also choose to fracture only rectilinear pin shapes. To do this, you can use the `AbstractFractureRectilinearShapes` option of the `absAbstract` SKILL function.

```
absSetBinOption("Block" "AbstractFractureRectilinearShapes" "true")
```

Important

If you are using the *Create boundary pins* or *Create ring pins* option on the *Adjust* tab and Abstract Generator has stretched a pin, then that pin is not fractured by this option. For example, if you have boundary pins for power nets but not for signal nets, only the power nets are fractured.

- *Fracture blockages*: This option controls whether Abstract Generator creates each blockage shape as a set of maximum rectangles or as a polygon. This option is available for all bins.

  When this option is not selected, which is the default, Abstract Generator creates each blockage shape as a polygon. This setting is recommended if the abstracts are to be routed using the IC Shape-Based Router.
  If you select this option, Abstract Generator creates each blockage shape as a set of maximum rectangles.

  During blockage generation, Abstract Generator converts ellipses, circles, and donuts into polygons before fracturing the blockages. By default, 64-point polygons are generated, which are snapped to the nearest manufacturing grid. You can overwrite this setting by using the following:

  ```
  absSetOptionabsSetOption("numOfPointsInEllipseOrDonut"
  "number_of_points")
  ```

  Example:

  ```
  absSetOption("numOfPointsInEllipseOrDonut" "64")
  ```

Setting Up Options for 45-Degree Geometry

LEF 5.8 supports the routing of 45-degree geometries. Abstract Generator provides support for preserving the 45-degree geometries as it is. By default, the conversion of 45-degree geometries to rectilinear ones is disabled. The options to convert a 45-degree geometry into

© 2020 All Rights Reserved.

stair-stepped rectangles are available only if one or both of *Fracture pins* and *Fracture blockages* options are active.

Setting the Type of Stair Step Coverage

The *Stair-step coverage* option controls the degree to which stair-step blockages cover the 45-degree geometry. There are three settings available for this option: *full*, *truncate*, and *partial*.

- With full stair-step coverage, the 45-degree geometry is completely obscured by blockage rectangles as shown below:

  This ensures that no real violations are created.

- With truncate stair-step coverage, the blockage rectangles are fully enclosed by the original 45-degree geometry as shown below.

© 2020 All Rights Reserved.

You might use this setting when there is another geometry close enough to cause violations with any stair-steps that extend outside the 45-degree edges.

Note: There is a considerable risk of the router being able to create routes that would not violate the abstract, but would violate the real 45-degree geometry.

- With partial stair-step coverage, the blockage rectangles only partially cover the 45-degree geometry as shown below.

This frees up more routing resources for the PR tools but because some of the real 45-degree geometry is exposed, there is an increased risk of violations.

Tip

In some situations, the stair-stepping procedure can create rectangles which introduce violations with neighboring geometry. This effect can be minimized by using a smaller step size or by using *truncate* rather than *partial* or *full* settings. Additional problems can be introduced, however, by using the *truncate* setting. The 45-degree geometry might be inadequately covered, and violations between wiring and 45-degree geometry would go undetected because of the modelling in the abstract. Again, using smaller stair-step distance can minimize this.

Setting the Width of the Stair Step

In the *Stair-steps width* field, specify the width of the blockage geometry to be created. Enter a value greater than zero. There is no minimum width restriction. The default value is half the `minSpacing` of the `Metal1` layer. If the `minSpacing` of `Metal1` layer is not found in the technology data, then the default value is set to `5`.

© 2020 All Rights Reserved.

Site Tab

To access the *Site* tab, select *Flow – Abstract*, then click the *Site* tab in the *Running step Abstract* form.

Sites tell the placer where it can place cells. All cells located in a particular bin share a common set of properties: symmetry, class (for example, pad or core), width, and height. These properties are used by the placer to assist in the placement process.

These cell properties are defined in a site object, and all cells within the same bin are associated with the same site. The site itself has a name, and the cells in the LEF file contain a site name field that holds the name of the associated site. You can create sites for the *Core, IO, Corner* bins.

In addition to the site names defined in the technology file, for CORE cells, the <AUTO> option is available in the Site Name cyclic field.

In addition to the site names defined in the technology file, for IO/Corner cells, the <AUTO> and <NONE> options available in the Site Name cyclic field.

© 2020 All Rights Reserved.

Behavior of the <AUTO> option:

- If the layout has an existing site definition, then no default site is created. The site definition from the layout is retained in the final abstract view and is dumped in the LEF file.
- If the layout does not have any site definition, a new default site definition is created and is dumped in the LEF file.

Behavior of the <NONE> option:

- Any existing site definition is removed from the abstract view. So, no site definition is dumped in the LEF file.
- If the layout does not have any site definition, then no new site definition is created. So, no site definition is dumped in the LEF file.

Note: The <NONE> option is not available for CORE cells because CORE cells must have a site definition in LEF file.

© 2020 All Rights Reserved.

Specifying a Site Name

In the *Site name* cyclic field, select a site name defined in the technology library from the available list to be applied to all cells in the current bin. Alternatively, select the *blank* option and allow Abstract Generator to select an appropriate site based on cell property settings.

The name of the site will appear in the LEF file exported from Abstract Generator. For any site to appear in the LEF generated from Abstract Generator, it must be defined in the technology library.

Defining a Site

In the *Define new site name* field, specify the name of a new site to be used by all cells in the current bin. If the site does not already exist, Abstract Generator defines it in the technology file. An entry in the *Define new site name* field overrides any setting in the *Site name* field.

Leaving this field blank automatically instructs Abstract Generator to use the site selected in the *Site name* field.

Both *Site name* and *Define new site name* fields associate cells with a single site.

© 2020 All Rights Reserved.

Calculating Site Pattern

For LEF 5.8, Abstract Generator can derive site patterns from the layout of gate-array standard cells. Abstract Generator provides this support only for gate-array cells with a single level of hierarchy. This site pattern calculation feature is available only for the *Core* bin.

To activate this feature, select the *Calculate site pattern for gate-array cells* option. This option works exclusive of the single site creation fields on the *Site* tab.

For valid site pattern calculation, the selected sub-cellviews (building blocks of gate-array cells) should be either associated to a predefined site or mapped to an already existing site. You can map the sub-cellviews to sites in the *Sub-Cellview to Site Mapping* table.

The *Add* button becomes active when you select the *Calculate SITE pattern for Gate Array Cells* option. Click *Add* to create the mappings. Specify the sub-cellview name in the *Sub-Cellview* column. In the *Site* column, the drop-down list displays the already existing sites. Select one from the list to map the selected sub-cellview.

Note: If a gate-array sub-cellview is itself associated with a site pattern, Abstract Generator preserves the pattern and exports it to LEF.

© 2020 All Rights Reserved.

Overlap Tab

To access the *Ovelap* tab, select *Flow – Abstract*, and then click the *Overlap* tab in the *Running step Abstract* form.

The overlap boundary is a second, more detailed boundary used by placement tools to test whether cells overlap. Unlike the PR boundary, which is always rectangular, the overlap boundary can be a rectilinear polygon. This helps the placement tools achieve better placement density for certain types of standard cells and rectilinear blocks. The options on this tab control whether Abstract Generator is to update the overlap boundary geometry.

Creating an Overlap Layer in OpenAccess

Because there is no concept of an overlap layer in OpenAccess, Abstract Generator creates the overlap shape on the PR boundary layer. If you export LEF for the cellview, however, the overlap shape is correctly written to the `OVERLAP` section.

Important

If you create or update the PR boundary by using the options on the Boundary Tab and then create an overlap layer by using the options on the *Overlap* tab, the PR boundary is destroyed and replaced with the overlap shape.

© 2020 All Rights Reserved.

Creating Overlap Boundary

You can use the *Create overlap boundary* option to specify whether Abstract Generator should create an overlap boundary. This option is a cyclic field and you can choose one of the following values:

- *always*: If you choose this setting, Abstract Generator always creates an overlap boundary. Any existing overlap layer geometry is overwritten.
- *as needed*: If you choose this setting, Abstract Generator creates an overlap boundary if it is not already present or if it is present and was previously calculated by Abstract Generator.
- *off*: If you choose this setting, Abstract Generator never creates an overlap boundary.

Specifying Layers and Geometries for the Overlap Boundary

In the *Using geometry on layers* field, you can specify the layers to be used in the overlap boundary calculation. The boundary is drawn so that it encloses all the geometry found on these layers.

Note: If you specify only a layer name instead of a layer-purpose pair, Abstract Generator considers all the purposes on that layer except the ones specified in the Ignored Purpose(s) field on the *General* tab (*File – General Options*).

Fine Tuning the Overlap Boundary

You can use the following options to specify information about the size and shape of the overlap boundary:

- *Size factor to apply*: In this field, specify (in microns) the value by which an overlap layer is to be increased. This option is enabled when the *Create overlap boundary* option is set to *always* or *as needed*. The default values depend on the current bin. The default for the *Block* bin is `0.0`; the default for the *Core* bin is half the minimum separation defined for the `metal1` layer.

  You can use this option to ensure that the placement of cells with abutting overlap layers does not lead to geometry contained in these cells causing violations.

  You are most likely to need this option when computing the overlap layer for standard cells that are placed automatically and abutted in rows. Blocks are abutted only rarely and should not require you to use this option.

- *Smooth factor to apply*: The value you specify in this field controls the shape of the overlap layer that is produced. This option is enabled when the *Create overlap*

© 2020 All Rights Reserved.

boundary option is set to *always* or *as needed*. The default values depend on the current bin. The default for the *Block* bin is `100` microns; the default for the *Core* bin is `0`.

You can use this option to create less complex overlap boundary shapes, typically when processing blocks. The value you supply is the maximum distance that you do not want to be cut out of the overlap layer. Abstract Generator takes the computed overlap boundary and removes any cut-outs with a maximum dimension less than the specified distance.

Example

Consider the following block. The PR boundary has been indicated with a broken line (`--`), and the outline of the geometry has been shown using the backslash character (`\\\`). This area marked by the backslashes indicates the area of the cell that contains shapes on the layers that will be used to create the overlap layer. (This area would become the overlap layer if the *Smooth factor to apply* was set to `0`).

When computing the overlap layer, it is likely that you want to retain the cut-out in the top left corner as free space. To do this, you must specify the value for the *Smooth factor to apply* option as `d`, where `d` is the distance in microns *slightly less* than the minimum dimension of

© 2020 All Rights Reserved.

the cut-out. Abstract Generator removes all other free space between the initial overlap boundary and the PR boundary but keeps the region in question.

© 2020 All Rights Reserved.

Retaining Layout Information in the Abstract View

Abstract Generator allows you to save layout details along with the pin shapes and blockages in the final abstract generated. To retain the layout information, you can use the `AbstractRetainLayout` option of the `absAbstract` SKILL command.

```
absSetBinOption("Block" "AbstractRetainLayout" "true")
```

By default, the `AbstractRetainLayout` option is set to `false`, so the final abstract contains the adjusted pin shapes, blockages, and boundary based on the options specified. All other shapes or data in the layout cell-view is deleted.

To include the layout information in the abstracts, set the `AbstractRetainLayout` option to `true`. The layout information along with the pin shapes and blockages will be written to the abstract view. You can change the name of the abstract view in the Views Tab of the *General options* dialog box (*File – General options)* by specifying the name in the *Abstract view name* field.

Alternatively, you can use the `absSetOption` SKILL command line to change the abstract view name.

```
absSetOption("ViewAbstract" "layout.pin")
```

You can use the `AbstractRetainLayout` option to create pins in your layouts so that they can be used in the Virtuoso® XL layout editor environment. If you do not want blockages in

© 2020 All Rights Reserved.

these layouts, do not specify any layers in the *Layer Assignment for Blockages* table on the *Blockage* tab of the Abstract step.

Creating Fill Blockages Automatically on Specific Layers

All detailed or cover blockages created in Abstract Generator are exported under the `OBS` section and all fill blockages under the `DENSITY` section of an LEF file. However, you can create fill blockages automatically on specific layers based on the detailed blockages generated in Abstract Generator.

By default, fill blockages are generated when the *Calculate metal density* option is turned on under the *Density* tab in the Abstract step. The following options can be set in the `abstract.replay` file or the `.abstract.options` file for creating fill blockages.

```
absSetBinOption("Core" "AbstractDensity" "true")

absSetBinOption("Core" "AbstractDensityLayers" "Metal1 Metal2
Metal3")
```

You can use the `AbstractPostHook` function to automatically create fill blockages on specific layers. This function is triggered after the creation of the abstract view, and it uses the point lists of the detailed or cover blockages that are generated. The fill blockages are then created on layers as per the `LayersList` variable defined in the SKILL code. You can also change the `blockageType` option to create other blockages such as feedthru, slot, and screen in the abstract view.

© 2020 All Rights Reserved.

Example: Type the following SKILL `AbstractPostHook` code at the `abstract` prompt.

```
procedure(AbstractPostHook(binName libName cellName abstractViewName)
let((cv LayersList blockageType RoutingBlockages)
cv = dbOpenCellViewByType(libName cellName abstractViewName "maskLayout"
"a")


LayersList=list("Metal4" "DIOdummy")   ;; change the list of layers on
which extra blockages are to be created
blockageType="fill"                    ;; Change the blockage type to any
fill, screen, feedthru etc.


RoutingBlockages=setof(x cv~>blockages x~>type=="routing")


foreach(blockage RoutingBlockages
    foreach(layer LayersList
           printf("Creating new Fill Blockage from layer = %s to layer =
%s at %L\n"
            blockage~>layer layer blockage~>points)
            dbCreateLayerBlockage(cv layer blockageType blockage~>points)
    );foreach
);foreach
dbSave(cv)
dbClose(cv)
) ;let
) ;procedure
```

Bus Annotation in Abstract Generator

Bus Annotation for Abstract Generator improves the interoperability between Virtuoso and Innovus. User can directly import the library elements created inside Abstract Generator to Innovus.

In OpenAccess, cellview data of a library element created by applications such as Abstract Generator (AG) usually consists of explicit scalar bits (A<0>, A<1>) corresponding to a vector bus terminal (A<0:1>). This is because pins and corresponding pinFigs have to be created on each bit with specified layer and shape for physical connectivity.

However, when OA cellview of library element is used by applications like Innovus and verilog2oa, as shown in the following two figures, the order of bits on a bus, which can be

© 2020 All Rights Reserved.

either ascending (0:1) or descending (1:0), is important. This cannot be inferred from scalar bits present in the OA data of the library element.

For this flow to work, run Verilog Annotate on the library elements before data is used in Innovus. The verilogAnnotate annotates the vector bus terminal information on the OA cellview of the library element. Running verilogAnnotate requires a corresponding interface definition in verilog format of the module corresponding to the library element. For each library element, a verilog stub file is created as described in the following figure. The verilog stub file is created from corresponding schematics and symbol definitions by using the Create –

© 2020 All Rights Reserved.

Cellview – From Cellview commands followed by Verilog Annotate. However, creating verilog stubs and running verilogAnnotate is a time consuming process.

In Virtuoso XL, while creating an abstract, an input logical view option can be used to specify the symbol, schematic, verilog, or .liberty (.lib) from which bus definition is extracted. The vector terminal information is directly added to the OA cellview of the library element by the

© 2020 All Rights Reserved.

Abstract Generator engine and there is no need to generate the verilog stub file and run verilogAnnotate.

Supporting EXCEPTPGNET LEF Attribute

The EXCEPTPGNET lef attribute is mapped to the OA attribute allowPGNet of oaLayerBlockage object. To support the EXCEPTPGNET LEF attribute in LEF5.8, the following bin specific option has been added in Abstract Generator.

```
absSetBinOption( "Core" "AbstractBlockageEXCEPTPGNET" "Metal1 Metal2")
```

When the AbstractBlockageEXCEPTPGNET option is set as specified in the above example, then all the blockages generated by Abstract on Metal1 and Metal2 will have EXCEPTPGNET lef attribute in the lef file.

This option is set for the blocks that want to block routing on a given layer except for power and ground routing. This is usually true for analog or memory blocks that are sensitive to noise, and do not want to allow signal routing on the layers immediately above the block. However, the intention is to allow power routing on the layers.

© 2020 All Rights Reserved.

Supporting Layout Dual View

A layout dual view contains both abstract and layout information. Innovus requires Abstract Generator to generate a partially annotated layout dual view. As a result, Abstract Generator annotates the required abstract information in the layout view and will not generate a separate abstract cellview. The following figure shows how a layout dual view is used in the Innovus flow.

Each of the three run steps of Abstract Generator—*Pins*, *Extract*, and *Abstract*—can generate the layout dual view. However, you need to run only the *Pins* step because the antenna, extraction, and blockage information (generated by Extract and Abstract steps) is not required in the layout dual view. When you run the *Pins* step, Abstract Generator saves a valid view of pins along with the layout dual view so that the *Pins* view can be used by the *Extract* and *Abstract* steps, if required.

Advantages of Layout Dual View

The advantages of layout dual view are as follows:

- Innovus no longer needs to maintain the abstract and layout views separately. This helps overcome the problem of maintaining two similar views for layouts that are under development.
- While routing, the router can directly use the layout dual view without the need to master and re-master the layout. This helps increase the productivity.

© 2020 All Rights Reserved.

Annotating Abstract Data in Layout Dual View

The abstract data that is annotated in the layout dual view is as follows.

- **prBoundary**

  The prBoundary of the layout dual view is updated if it already exists in the layout view; otherwise, it is created based on the *Create boundary* options on the *Boundary* tab of Abstract Generator. To know more, refer to Specifying the Creation of the PR Boundary.

- **cellType**

  If the cellType is already specified for the layout, Abstract Generator does not modify the cellType in the layout dual view. If the cellType attribute is not specified, Abstract Generator automatically sets it depending on the bin in which the cell is processed. To know about the default behavior of Abstract Generator for specifying the cellType, refer to Distributing Cells Based on Type.

- **symmetry**

  Abstract Generator uses the following sequence to determine the symmetry of the current cellview:

  - The *Symmetry Type* attribute in layout cellview is considered. For more information about setting this attribute, see Edit Cellview Properties Form.
  - If the *Symmetry Type* attribute is not set in layout cellview, then the property with the name *Symmetry* on layout cellview is considered.
  - You can overwrite the above values by selecting a value in the *symmetry* field on the Cell Properties form. For more information about this form, see symmetry.

- **timestamp**

  The timestamp of the annotated abstract data is updated in the layout dual view. This is done to verify whether or not the abstract data in the layout dual view is up to date.

- **Cover blockage**

  The cover blockage is created for prBoundary in layout dual view with the value provided by the *AbstractTopLayerCoverBlockage* option.

- **Pin Annotation**

  Abstract generator enables you to create pins from the text labels by using the *Map text label to pins* option in the Pins step. You can also create pins from the text labels by setting the `PinsTextPinMap` option, as follows:

```
absSetBinOption("Core" "PinsTextPinMap"`(M1 (M1 label))")
```


© 2020 All Rights Reserved.

The new pins created by Abstract Generator based on the pin text label map are annotated in the layout dual view.

If the layout has existing pins and pin text label map is set, only the new pins added by Abstract Generator are annotated in the layout dual view. If the layout has existing pins and pin text label map is not set, no new pins are created and the existing pins are preserved in the layout dual view. To know more about the pin options, refer to Pins Step.

Abstract Generator options for Layout Dual View

For generating the layout dual view, the following two new options have been added to Abstract Generator.

- **AnnotateLayoutDualView**

  To enable Abstract Generator to generate the layout dual view, set the `AnnotateLayoutDualView` option to `TRUE`. If this option is set to `TRUE`, Abstract Generator annotates the abstract data in the layout view to generate the layout dual view. By default, this option is set to `FALSE`.

  ```
  absSetOption("AnnotateLayoutDualView" "true")
  ```

- **AbstractTopLayerCoverBlockage**

  This option specifies the top metal layer for cover blockage in the layout dual view. This new option has been added because the existing cover blockage options have different use model, which may lead to confusion.

  ```
  absSetBinOption("Core" AbstractTopLayerCoverBlockage" "Metal5")
  ```

Support for Copying Mixed Signal Routing Constraints from Layout to Abstract View

In a layout design, you can create mixed signal routing constraints on a block using the Virtuoso Constraint Manager. However, when an abstract is created for a block from a layout design, the abstract view does not contain the constraint information of the block.

Abstract Generator provides support for copying mixed signal routing constraints from the layout view to the abstract view. In this release, the following mixed signal routing constraints are supported.

- ciDiffPairType
- ciShieldingType
- ciNetBusType

© 2020 All Rights Reserved.

- ciSymmetryType
- ciProcessOverrideType
- ciMatchedLengthType

You can copy the option signal routing constraints from the layout view to the abstract view, either while a new abstract is being generated or to an existing abstract, as explained below:

- Copying constraints to new abstracts during abstract generation

  To copy mixed signal routing constraints to the new abstracts that are created, set the `CopyMSRoutingConstraints` option to `TRUE`. For more information, see CopyMSRoutingConstraint.

  In addition, for a constraint to be copied to the abstract view, the following three conditions must be satisfied:

  - The routing constraints must be present in the top-level layout view
  - The routing constraint must be present in a net that is part of the abstract
  - The routing constraint must have the Hierarchical Scope attribute set to ‘above’

  Note: Hierarchical scope is an attribute of a routing constraint that can have the following values: local, below, and above. The value ‘local’ is always set for a routing constraint.

  Important

  By default, the value of the Hierarchical scope attribute is set to local. Therefore, ensure that you set it to above before you create the abstract from the layout.

- Copying constraints to existing abstracts without regenerating the abstracts

  You can add mixed signal routing constraints to existing abstracts in VLS XL and VLS GXL. To add constraints, open the existing abstract view in the Virtuoso Layout window and use one of the following methods:

  - Load the constraints from the specified connectivity source view by choosing *Connectivity – Update – Layout Constraints.* For more information, refer to Updating Layout Constraints in the *Virtuoso Layout Suite XL User Guide.*
  - Copy the constraints from a layout view by using the *Load Physical View* command. For more information, refer to Load Physical View in the *Virtuoso Layout Suite XL User Guide*.
  - Add constraints to an existing abstract by using the Routing constraints available in the Constraint Creation list in the Constraint Manager. For more information, refer to

© 2020 All Rights Reserved.

Constraint Manager Assistant in the *Virtuoso Unified Custom Constraints User Guide.*

Suppressing Messages Issued by Abstract Generator

Use the `suppressMessage` option to suppress INFO and WARNING messages issued by Abstract Generator.

```
absSetOption(suppressMessage <id1> <id2> <id3> …)
```

This option suppresses the messages with the mentioned message IDs. ERROR messages are not suppressed. If any ERROR message ID is mentioned in `suppressMessage`, then a warning is issued stating that this is an ERROR message and will not be suppressed. For every suppressed message, an INFO message is displayed that this message is suppressed and will not be shown. This INFO message is displayed only once. At the end of the Abstract step, a table shows the messages that have been suppressed in the current run, and this information is displayed for every cell.

For more information about messages issued by Abstract Generator, see Abstract Generator Error and Warning Messages.

© 2020 All Rights Reserved.

6

Integrated Abstract Generator

This chapter describes how to use the integrated Abstract Generator to generate abstracts while you are working on layout designs in the Virtuoso design environment.

The integrated Abstract Generator offers a subset of the standalone Abstract Generator tool, with the minimum and mandatory GUI options. It is presented as a form in which all the abstract steps are available, segregated through progressive disclosure buttons, with the most frequently used options as part of main Abstract Generator flow steps.

This chapter includes the following sections:

- Launching the Integrated Abstract Generator
- Customizing the Interface
- Generating Abstracts
  - Loading Values in the Abstract Generator Form
  - Running the Abstract Generator Steps
  - Viewing Results
- Environment Variables
- Form Descriptions

© 2020 All Rights Reserved.

Launching the Integrated Abstract Generator

The integrated Abstract Generator in the Virtuoso design environment is a simple interface to generate abstracts for one or more cellviews.

To launch the integrated Abstract Generator interface, do the following:

➡ In the CIW, choose *Tools - Abstract Generator*.

The Abstract Generator form is displayed.

Corresponding SKILL API: iagGenAbstract

Customizing the Interface

To keep the user interface as simple as possible, when you launch the integrated Abstract Generator interface, only the *Design Information* section is visible by default. Sections corresponding to the *Pins*, *Extract*, and *Abstract* steps are hidden. Set the following environment variables to `t` to display these sections by default.

Pins: showPinsSection

© 2020 All Rights Reserved.

Extract: showExtractSection, showAntennaOptions, showExtractGspecTables

Abstract: showAbstractSection, showOverlapBoundaryOption, showPinGroupingOptions

© 2020 All Rights Reserved.

When you set these environment variables to `t`, the Abstract Generator form looks like this:

© 2020 All Rights Reserved.

Generating Abstracts

You can perform the following operations during the abstract generation process.

- Loading Values in the Abstract Generator Form
- Running the Abstract Generator Steps
- Viewing Results

Loading Values in the Abstract Generator Form

To initialize the fields in the Abstract Generator form, do the following:

1. Select a library.

   The library list is read in from the `cds.lib` file located in the run directory, and libraries are listed in an alphabetical order. When you select a library, the *Cell(s)* list is automatically populated.

2. From the *Cell(s)* list, select the required cells. Use one of the following selection methods:

   - Select one or multiple cells by clicking their names.
   - Select multiple consecutive cells by clicking a cell and dragging the cursor across all the cells to be selected without releasing the mouse button. Release the mouse button after all the required cells have been selected.
   - Select all cells by using the `Ctrl+A` bindkey.

   The number of cells selected is displayed in the status bar.

   To deselect cells:

   - Click the individual cells to be deselected.
   - To deselect all the cells at one go, use the `Ctrl+D` bindkey.

   The *Filter* Cells field lets you filter out cells based on their names. For example, type `top` to filter out cells that contain the string in their name.

3. In the *View* field, specify the view name in which you want to create abstracts.

4. From the *Design Type* drop-down list, select a design type. Design type indicates the type of cells to be processed in this abstract run: *Block* (default), *Core, IO*, and *Corner*.

© 2020 All Rights Reserved.

5. (Optional) Specify a *Top Metal* layer. If you specify a top metal layer, the layers listed in the *Extract* and *Abstract* steps are automatically restricted to the specified top metal layer. This means that abstract shapes are generated only for objects on the specified layer and its lower layers, and not for any others. For example, if *Top Metal* is set to `M3`, then the other layer options are reset to include only layers `M3` and its lower layers. Abstract shapes are generated only for objects on these layers.
6. In the *Options File* field, specify the path from where Abstract Generator loads the options.
7. Click *Load Options*.

Loading the Options File

After you click *Load Options*, Abstract Generator loads the options file in the following order:

1. The options file (`.abstract.options`) from the `<current working directory>/.abstract/<library name>` directory is loaded. If not available in the above location, Abstract Generator reads the file first from the attached technology library directory and then from the design library directory.
2. The file specified in the *Options File* field in the Abstract Generator form is loaded.

If the same options are available in the next file loaded, the options are overwritten accordingly.

Running the Abstract Generator Steps

When all the options are loaded, the Abstract Generator form features separate sections for each of the abstract generation steps.

Pins Step

Pins is the first step of the abstract generation process. The showPinsSection environment variable controls the visibility of the *Pins* section. You can use the progressive disclosure arrow to reveal or hide the options.

© 2020 All Rights Reserved.

The integrated Abstract Generator performs the following operations during the *Pins* step:

- **Preserve Local Blockages:** This option controls how Abstract Generator processes local blockages present in the layout view.
  - When unselected (default), Abstract Generator removes any local blockages, and proceeds as if they had never been specified in the DEF file.
  - When selected, Abstract Generator preserves the local blockages, along with their attributes, from level 0 to level 32.
- **Create Pins From Labels:** This option creates physical pin shapes corresponding to the geometry overlapping the text labels. Additionally, specify the text labels in the *Label to Pin Map* field. Abstract Generator creates pins by mapping overlapping geometries to text labels specified in this field.

Extract Step

Extract, the second stage in the abstract generation process, involves extraction of new pins by tracing the physical connections starting from pins created in the *Pins* step. Antenna numbers are also calculated based on the options selected.

The showExtractSection, showAntennaOptions, and showExtractGspecTables environment variables control the visibility of various options in the *Extract* section. You can use the progressive disclosure arrow to reveal or hide the options.

The integrated Abstract Generator performs the following functions during the *Extract* step:

- **Extracts Connectivity Information**

© 2020 All Rights Reserved.

- Choose whether you want to extract connectivity information for *Signal*, *Power*, or both types of nets.
- *Connectivity* specifies the connectivity between layers. You can specify the layer connectivity without having to add via definitions in the technology file. The format for entering a value in this field is to list the layers as a number of entries; each entry being a combination of two or three layers.

  `(layer1 layer2 layer3) (layer4 layer5 layer6) ... (layer7 layer8) ...`

  Example:

  `(M1 M2 VIA1)(M2 M3 VIA2)(M4 M5)`

Calculates Antenna Information

- Choose *Calculate Antenna* to create process antenna information for custom blocks and standard cells.
- Use the *Gate* and *Drain* text boxes to assign layers to antenna regions to identify the gate and drain regions and to enable antenna calculation.

  The *Gate* region is identified as the intersection of the first `Poly` layer with all diffusion layers. **Example:** (`Poly and Active`). Here, `Poly` is the gate identification layer.

  The *Drain* region is identified as the `andnot` of the first diffusion layer with all `Poly` layers. **Example:** (`Active andnot Poly`). Here, the diffusion layer acts as the drain identification layer.

  Note: The integrated Abstract Generator form supports only one Oxide model. This means that only the first diffusion layer is identified as the drain region. If there is more than one diffusion layer in the technology file, then the drain identification must be set through the options file. In this case, the form shows only the first drain region identification setting.

Extracts Connectivity Information for Signal and Power Nets

Signal Nets:

- *Signal Layers* lists all signal layers to be extracted. Select *Use Signal GSpec for Power GSpec* to extract the geometry specifications for these layers.
- *Pin Layers* lists all the layers on which pins need to be created for signal nets.

Power Nets:

© 2020 All Rights Reserved.

- *Power Layers* lists all power layers to be extracted. Select *Use Signal GSpec for Power GSpec* to extract the geometry specifications for these layers.
- *Pin Layers* lists all the layers on which pins need to be created for power nets.

Note: The *Signal Layers* and *Power Layers* list is controlled by the *Top Metal* layer specified in the *Design Information* section. For example, if *Top Metal* is set to `M3`, then only layer `M3` and its lower layers can be listed in the *Signal Layers* and *Power Layers* text boxes. Therefore, Abstract Generator can be run only on these layers.

Abstract Step

Abstract is the third, and final step of the abstract generation process. The showAbstractSection environment variable controls the visibility of the *Abstract* section. You can use the progressive disclosure arrow to reveal or hide the options.

The integrated Abstract Generator performs the following operations during the *Abstract* step:

Creates Boundary Pins

The *Create Boundary Pins* option lets you create square pins at the boundary from the extracted pin geometries. Choose whether you want to create boundary pins for *Signal* nets, *Power* nets, or both.

Use the *Create Overlap Boundary* option to specify whether a rectilinear PR boundary (overlap) should be generated during abstraction. If the `.abstract.options` file is loaded, the layers specified in the `AbstractOverlapLayers` section are used to create the overlap boundary.

Groups Signal and Power Nets

The *Signal Grouping* and *Power Grouping* options control how the pin geometries are grouped for corresponding signal and P/G nets. This is useful for blocks with numerous

© 2020 All Rights Reserved.

signal and/or power pins on different layers. *Signal Grouping* and *Power Grouping* are cyclic fields that provide the following options:

Single	Creates a single pin geometry for all net shapes.
Separate	Creates separate pin geometries for each net shape.
Overlap	Forms pin geometries from groups of overlapping net shapes.

Tip

Set a value for this option after you have selected the corresponding *Create Boundary Pins* option.

- **Adds Cover Blockages**

  Use the *Cover Blockages* text box to specify the layers to be considered while creating cover blockages. If a layer is removed from this list, then the layer is considered while created detailed blockages.

- **Cuts Blockages around Pins**

  Pin CutOut lists the layers on which pin cutouts need to be created. Pin cutouts are applicable for all types of blockages: cover, detailed, and shrink.

  Note: The list of layers is limited by the *Top Metal* specified in the *Design Information* section.

- **Generates Rectangular Geometries for Blockages and Pins**

  The *Fracture Blockages* and *Fracture Pins* options let you generate only rectangular geometries for pins and blockages in the final abstract view. When the options are selected, all generated pins and blockages in the abstract view are fractured to generate only rectangular geometries. Otherwise, their shapes are retained.

Viewing Results

After updating values in the above sections, click *Generate Abstract* to run Abstract Generator. The Abstract Log window is displayed. You can use this window to check whether the results are satisfactory. You can also use the Cell Report window to check the status of the cells that were included in the abstract generation process.

© 2020 All Rights Reserved.

Viewing the Cell Report

To view the cell report,

➡ In the Abstract Generator form, choose *Tools - Show Cell Report*.

The Cell Report window is displayed.

CellName column displays a list of all the cells that are selected in the *Cell(s)* field in the *Design Information* section of the Abstract Generator form.

Status column displays the status of the cells, green indicates no errors or warnings, orange indicates warnings, and red indicates errors during abstract generation.

Errors displays the number of errors that were reported during abstract generation.

Warnings displays the number of warnings that were reported during abstract generation.

Report displays the *View* button against every cell. Click the button to view detailed information about the abstract generation process, including the time taken for abstract generation, types of blockages created on different layers, and the complete list of errors and

© 2020 All Rights Reserved.

warnings that were reported.

Library name: design
Cell name: test_bin1
View name: layout

Time taken by the abstract generator processes:

Process	Real Time (sec)	CPU Time (sec)
Pins	0.018	0.010
Extract	0.006	0.010
Final pin creation	0.002	0.000
Blockage creation	0.010	0.010
Abstract view creation	0.141	0.090

Extracted nets: 1
Extracted shapes: 1

Types of blockages created on different layers:

Cover blockage: Metal1
Detailed blockage: Metal2
Shrink-wrap blockage: NIL

Warning(s) :

Message ID	Occurrences	Message
15070	1	The test_bin1 cell has no power terminals.
15071	1	The test_bin1 cell has no ground terminals.
15073	1	The test_bin1 cell has no output terminals.
11026	1	Bus annotation cannot be completed for the specified abstract because the logical view for cell test_bin1 is invalid.

Warning count: 4

Viewing Log Information

The Abstract Log window displays runtime information, including all the information, warning, and error messages issued by Abstract Generator during the current session.

To view log information,

➡ In the Abstract Generator form, choose *Tools - Show Abstract Log*.

© 2020 All Rights Reserved.

The Abstract Log window is displayed.

The log information is also written to the `CDS.log` file.

Viewing the List of Cells

The Selected Cells window displays a list of all the cells that are selected in the *Cell(s)* field in the *Design Information* section of the Abstract Generator form.

To view the selected cells,

© 2020 All Rights Reserved.

➡ In the Abstract Generator form, choose *Tools - Show Selected Cells*.

The Selected Cells window is displayed.

© 2020 All Rights Reserved.

Environment Variables

This section provides information on the environment variables that can be used with the integrated Abstract Generator. Only the following environment variables are supported for public use:

- optionsFile
- showAbstractSection
- showAntennaOptions
- showExtractGspecTables
- showExtractSection
- showOverlapBoundaryOption
- showPinsSection
- showPinGroupingOptions

© 2020 All Rights Reserved.

optionsFile

```
abstract optionsFile string "options_file_path"
```

Description

Specifies the path to the options file. The default value of this environment variable is an empty string (`""`).

GUI Equivalent

Command	*Tools – Abstract Generator – Design Information*
Field	*Options File*

Examples

```
envGetVal("abstract" "optionsFile")
envSetVal("abstract" "optionsFile" 'string "server1/abstract/Files/AG.options")
envSetVal("abstract" "optionsFile" 'string "")
```

Related Topics

Customizing the Interface

Loading the Options File

© 2020 All Rights Reserved.

showAbstractSection

```
abstract showAbstractSection boolean { t | nil }
```

Description

Determines whether the *Abstract* section is displayed by default when you open the Abstract Generator form. When set to `t`, the section is displayed. The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("abstract" "showAbstractSection")
envSetVal("abstract" "showAbstractSection" 'boolean t)
envSetVal("abstract" "showAbstractSection" 'boolean nil)
```

Related Topics

Abstract Step

Customizing the Interface

© 2020 All Rights Reserved.

showAntennaOptions

```
abstract showAntennaOptions boolean { t | nil }
```

Description

Determines whether the following antenna options are displayed by default when you open the *Extract* section of the Abstract Generator form:

- *Gate*
- *Drain*

When set to `t`, these options are displayed. The default value is `nil`.

GUI Equivalent

Command	*Tools – Abstract Generator – Extract* section
Field	*Gate* and *Drain*

Examples

```
envGetVal("abstract" "showAntennaOptions")
envSetVal("abstract" "showAntennaOptions" 'boolean t)
envSetVal("abstract" "showAntennaOptions" 'boolean nil)
```

Related Topics

Extract Step

Customizing the Interface

© 2020 All Rights Reserved.

showExtractGspecTables

```
abstract showExtractGspecTables boolean { t | nil }
```

Description

Determines whether the options for extracting connectivity information for signal and power nets are displayed by default when you open the *Extract* section of the Abstract Generator form. When set to `t`, the following options are displayed in the *Extract* section:

- *Signal Layers*
- *Pin Layers*
- *Power Layers*
- *Use Signal GSpec For Power GSpec*

The default value is `nil`.

GUI Equivalent

Command	*Tools – Abstract Generator – Extract* section
Field	*Signal Layers*, *Pin Layers*, *Power Layers*, and *Use Signal GSpec For Power GSpec*

Examples

```
envGetVal("abstract" "showExtractGspecTables")
envSetVal("abstract" "showExtractGspecTables" 'boolean t)
envSetVal("abstract" "showExtractGspecTables" 'boolean nil)
```

Related Topics

Extract Step

Customizing the Interface

© 2020 All Rights Reserved.

showExtractSection

```
abstract showExtractSection boolean { t | nil }
```

Description

Determines whether the *Extract* section is displayed by default when you open the Abstract Generator form. When set to `t`, the section is displayed. The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("abstract" "showExtractSection")
envSetVal("abstract" "showExtractSection" 'boolean t)
envSetVal("abstract" "showExtractSection" 'boolean nil)
```

Related Topics

Extract Step

Customizing the Interface

© 2020 All Rights Reserved.

showPinsSection

```
abstract showPinsSection boolean { t | nil }
```

Description

Determines whether the *Pins* section is displayed by default when you open the Abstract Generator form. When set to `t`, the section is displayed. The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("abstract" "showPinsSection")
envSetVal("abstract" "showPinsSection" 'boolean t)
envSetVal("abstract" "showPinsSection" 'boolean nil)
```

Related Topics

Pins Step

Customizing the Interface

© 2020 All Rights Reserved.

showOverlapBoundaryOption

```
abstract showOverlapBoundaryOption boolean { t | nil }
```

Description

Determines whether the *Create Overlap Boundary* option is displayed by default when you open the *Abstract* section of the Abstract Generator form. When set to `t`, the option is displayed. The default value is `nil`.

GUI Equivalent

Command	*Tools – Abstract Generator – Abstract* section
Field	*Create Overlap Boundary*

Examples

```
envGetVal("abstract" "showOverlapBoundaryOption")
envSetVal("abstract" "showOverlapBoundaryOption" 'boolean t)
envSetVal("abstract" "showOverlapBoundaryOption" 'boolean nil)
```

Related Topics

Abstract Step

Customizing the Interface

© 2020 All Rights Reserved.

showPinGroupingOptions

```
abstract showPinGroupingOptions boolean { t | nil }
```

Description

Determines whether the options for grouping signal and power nets are displayed by default when you open the *Abstract* section of the Abstract Generator form. When set to `t`, the following options are displayed in the Abstract section:

- *Signal Grouping*
- *Power Grouping*

The default value is `nil`.

GUI Equivalent

Command	*Tools – Abstract Generator – Abstract* section
Field	*Signal Grouping* and *Power Grouping*

Examples

```
envGetVal("abstract" "showPinGroupingOptions")
envSetVal("abstract" "showPinGroupingOptions" 'boolean t)
envSetVal("abstract" "showPinGroupingOptions" 'boolean nil)
```

Related Topics

Abstract Step

Groups Signal and Power Nets

Customizing the Interface

© 2020 All Rights Reserved.

Form Descriptions

This section describes the various menus and fields of the Abstract Generator form.

File menu includes the *Close* command to close the Abstract Generator form.

Tools menu includes the following commands:

Show Cell Report displays the status of the cells included in the abstract generation process.

Show Abstract Log displays log information of the current session.

Show Selected Cells displays a list of all the cells that are selected in the *Cell(s)* field in the *Design Information* section.

Design Information section includes various details that are specified in the design layout.

Library lets you specify the library containing the cells for which you want to create abstracts. The library list is read in from the `cds.lib` file located in the run directory, and libraries are listed in an alphabetical order.

Filter Cells lets you filter out cells based on their names. For example, type `top` to filter out cells that contain the string in their name.

Cell(s) lets you specify the cells for which you want to create abstracts. The *Cell(s)* list box displays the names of all the cells in the library selected in the *Library* drop-down list.

View lets you specify the view name in which you want to create abstracts.

Design Type indicates the type of cells to be processed in this abstract run: *Block* (default), *Core, IO*, and *Corner*.

Top Metal lets you specify the top metal layer. If you specify a *Top Metal* layer, abstract shapes are generated only for objects on the specified layer and its lower layers.

Options File lets you specify the path from where Abstract Generator loads the options.

Load Options loads the options from the specified options file.

Pins extracts pin information.

Environment variable: showPinsSection

Preserve Local Blockages controls how Abstract Generator processes local blockages present in the layout view.

© 2020 All Rights Reserved.

Create Pins From Labels creates physical pin shapes corresponding to the geometry overlapping the text labels.

Label to Pin Map lets you specify the text labels. Abstract Generator creates pins by mapping overlapping geometries to text labels specified in this field. This field is enabled only when the *Create Pins From Labels* check box is selected.

Extract extracts connectivity data and calculates antenna information.

Environment variables: showExtractSection, showAntennaOptions, showExtractGspecTables

Signal extracts connectivity information for signal nets.

Power extracts connectivity information for power nets.

Calculate Antenna creates process antenna information for custom blocks and standard cells.

Connectivity specifies the connectivity between layers. This field is enabled only when one the three check boxes, *Signal*, *Power*, and *Calculate Antenna*, is selected.

Gate indicates the intersection of the first `Poly` layer with all diffusion layers in the antenna region. This field is enabled only when the *Calculate Antenna* check box is selected.

Drain indicates the drain identification layer in the antenna region. It is identified as the `andnot` of the first diffusion layer with all `Poly` layers. This field is enabled only when the *Calculate Antenna* check box is selected.

Signal Layers lists all signal layers to be extracted. This field is enabled only when the *Signal* or the *Calculate Antenna* check box is selected.

Pin Layers lists all the layers on which pins need to be created for signal nets. This field is enabled only when the *Signal* check box is selected.

Use Signal GSpec For Power GSpec extracts geometry specifications for the power layers. This field is enabled only when the *Power* check box is selected.

Power Layers lists all power layers to be extracted. This field is enabled only when the *Power* check box is selected.

Pin Layers lists all the layers on which pins need to be created for power nets. This field is enabled only when the *Power* check box is selected.

Abstract creates the final shapes required by place-and-route tools, and fractures pin shapes into rectangles.

© 2020 All Rights Reserved.

Environment variables: showAbstractSection, showOverlapBoundaryOption, showPinGroupingOptions

Create Boundary Pins lets you create square pins at the boundary from the extracted pin geometries.

Signal creates boundary pins for signal nets.

Power creates boundary pins for power nets.

Create Overlap Boundary lets you specify whether a rectilinear PR boundary (overlap) should be generated during abstraction.

Signal Grouping controls how the pin geometries are grouped for corresponding signal nets. You can choose from *Single*, *Separate*, or *Overlap* (default).

Power Grouping controls how the pin geometries are grouped for corresponding power nets. You can choose from *Single*, *Separate*, or *Overlap* (default).

Cover Blockages lets you specify the layers to be considered while creating cover blockages.

Pin CutOut lets you specify the layers on which pin cutouts need to be created.

Fracture Blockages generates only rectangular geometries for blockages in the final abstract view.

Fracture Pins generates only rectangular geometries for pins in the final abstract view.

Generate Abstract generates abstracts for the selected cells.

Close closes the Abstract Generator form.

Help displays information about the form fields.

© 2020 All Rights Reserved.

7

Dynamic Abstract Generation

Dynamic Abstract Generation automatically creates abstracts for the Pcells based on a set of abstraction rules. The abstract data is stored in memory and is annotated to the Pcell submasters corresponding to the Pcells in the design. Because an annotated Pcell submaster contains abstract data annotated over the layout geometries, it is called the dual view. The dual view can be used directly by the router without the need to master and remaster the layout.

Note: Dynamic Abstract Generation is used by Virtuoso Space-Based Router and Create Wire.

Dynamic Abstract Generation has been developed specially for device-level routing— the blockage information that the abstracts contain helps avoid routing over devices. The following additional capabilities of Dynamic Abstract Generation also helps in routing:

- Creates an accurate place and route boundary (prBoundary)

© 2020 All Rights Reserved.

- Creates detailed blockages by default; also provides the option to create cover blockages
- Creates and resizes pins and preserves existing pins; pins can also be created from text labels
- Identifies strongly connected pins, weakly connected pins, and mustjoin pins
- Extracts nets by using the layer-geometry specification

Note: Applications using the Dynamic Abstract Generation API, need to check out the Layout XL license. It does not require a separate license.

Prerequisites for Dynamic Abstract Generation

By default, Dynamic Abstract Generation uses the `Wire` constraint group that is specified in the *Wire Editing* section of the Layout Editor options form. For more information about the `Wire` constraint group, refer to *Virtuoso Layout Suite L User Guide*.

The prerequisites for Dynamic Abstract Generation are as follows:

- Valid vias and valid layers must be defined in the specified constraint group. If valid vias and valid layers are not present in the constraint group, a warning message is displayed in the CIW stating that the required information be added in the technology file.
- The `minSpacing` and `minWidth` layer rules must be present in the technology file. During Dynamic Abstract Generation, the constraint groups are searched for layer rules in the following order:
  - `Wire` (user-defined) constraint group
  - `LefDefaultRouteSpec` constraint group by type
  - `LefDefaultRouteSpec` constraint group by name
  - `Foundry` constraint group

  For example, if the `minSpacing` value for *metal3* is required, the `minSpacing` value is first searched for in the `Wire` constraint group. If the `minSpacing` value is found in the `Wire` constraint group, it is used. If the `minSpacing` value is not found in the `Wire` constraint group, the `minSpacing` value is searched for in the `LefDefaultRouteSpec` constraint group first by type and then by name. If the `minSpacing` value is not found in the `LefDefaultRouteSpec` constraint group too, the `Foundry` constraint group is searched for the `minSpacing` value.
- Net and pin technology purpose must be present in the technology file. During Dynamic Abstract Generation, the net and pin purpose is used for net extraction and pin creation. If the net and pin purpose is not present in the technology file, Dynamic Abstract

© 2020 All Rights Reserved.

Generation adds the layer purpose pair information in the technology file, provided the write permission is available for the technology file.

- If an `alternate foundry` constraint group (AFCG) is defined in the `.cdsenv` file, Dynamic Abstract Generator replaces the `foundry` constraint group and uses the specified AFCG instead.

  You can specify an AFCG by setting the following environment variable in your `.cdsenv` file:

  ```
  cdba.layout AlternateFoundryCG string "AFCG_name"
  ```

Dynamic Abstraction Generation Use Model

In an OpenAccess design that contains Pcells, you can run Dynamic Abstract Generation manually by using the commands that can be accessed from the following three locations:

- *Tools – Dynamic Abstract Generation for Pcells*
- Navigator assistant (by using the context-sensitive menu). For example, you can use the Navigator assistant to run Dynamic Abstract Generation for Module Generator (Modgen) instances. To do this, select a Modgen instance in the Navigator assistant and select *Dynamic Abstract – Abstract Pcells* from the context-sensitive menu. To know more about Modgens, see Working with Module Generators.
- Layout canvas (by using the context-sensitive menu)

You can create abstracts for all the Pcells in the design or for only the selected Pcells, and then run the routing or wire editing commands. To create abstracts, you use either the default options or specify your own abstraction rules in the Abstract Pcells form. For more information, refer to Abstracting Pcells.

When you run the automatic routing or wire editing commands, Dynamic Abstract Generation is run in the background for all the Pcell instances based on the predefined abstraction rules.

If the routing results are satisfactory, the routing process is complete. Otherwise, you can modify the abstraction rules, such as modify layer information for blockages, as required and run the process until satisfactory results are obtained.

© 2020 All Rights Reserved.

The following flow diagram illustrates the Dynamic Abstract Generation process.

Note: By default, all Pcells are marked for abstraction. If a Pcell is unmarked for abstraction, the Pcell needs to be marked again for abstraction. For more information, refer to Marking or Unmarking Pcells for Abstraction.

Dynamic Abstract Generation Commands

You can use the Dynamic Abstract Generation commands to abstract, unabstract, highlight, mark, and unmark Pcells. To access the commands, in the Layout window, choose *Tools* –

© 2020 All Rights Reserved.

Dynamic Abstract Generation for Pcells. A submenu is displayed with the commands shown in the figure below.

- Abstracting Pcells
- Unabstracting Pcells
- Highlighting Unabstracted Pcells
- Marking or Unmarking Pcells for Abstraction

Video

For quick reference of Dynamic Abstract Generator commands, refer to the Using Dynamic Abstract Generator demonstration.

Abstracting Pcells

To generate an abstract of a Pcell, in the Layout window, choose *Tools – Dynamic Abstract Generation For Pcells – Abstract Pcells*. The Abstract Pcells form is displayed as shown in the figure below.

© 2020 All Rights Reserved.

Note: When you point to an option in the Abstract Pcells form, an info balloon that describes the purpose of the option is displayed.

Selecting Pcells

When the *All Pcells* option is selected in the Abstract Pcells form, abstracts are created for all the Pcells in the design. If you do not want to create the abstracts for all the Pcells, you can select the required Pcells in the layout canvas or from the Navigator assistant and ensure that the *Selected Pcells* option is selected. After you have run the command, you can select additional Pcells for abstraction, if required, and run the command again.

The abstracts are created for all Pcells that are selected, except the ones that are unmarked for abstraction. A warning message is displayed stating that some Pcells could not be abstracted because the Pcells were unmarked for abstraction.

A warning message is also displayed if abstract data already exists for a Pcell. To generate an abstract for such a Pcell, delete the abstract data by using the *Unabstract Pcells* command, and again abstract the required Pcells.

© 2020 All Rights Reserved.

Viewing the Abstract Status

To determine whether the abstracts were created successfully:

- View the blockages created at level-1in the layout canvas, as shown in the following figure.

Blockages visible for an abstracted Pcell

© 2020 All Rights Reserved.

- View the status icons associated with the various Pcells in the Navigator assistant. For more information on the status icons, refer to Navigator Assistant (Using the Context-Sensitive Menu).

Specifying Abstraction options

You use the *options* group box in the Abstract Pcells form to specify the settings for blockage generation and for loading or saving an options file.

Specifying Blockages

- Cover Blockage Layers

  Specify layers in the *Cover Blockage Layer*s field. This creates cover blockages for all layers and layer purpose pairs specified in the *Cover Blockage Layer*s field. For example if you specify `Metal1 Metal2 Metal3`, the cover blockage is created on all the three metal layers. By default, the *Cover Blockage Layers* field is empty.

© 2020 All Rights Reserved.

A warning message is displayed if the layers specified in the *Cover Blockage Layers* field are not complementary to the ones specified in the *Detailed Blockage Layers* field. For example, if you specify `Metal1` in the *Cover Blockages Layer* field and `Metal1 (Metal1 or Oxide)` in the *Detailed Blockage Layers* field, the following message is displayed:

Cover blockage layers (`"Metal1"`) are already part of detailed blockage layer list (`"Metal1" "Oxide"`). Specify cover blockage layer list such that cover blockage layers are complementary to detailed blockage layers.

- Detailed Blockage Layers

  Specify a layer-geometry specification in the *Detailed Blockage Layers* field. This creates detailed blockages for all layers and layer purpose pairs specified in the *Detailed Blockage Layers* field. The layer geometry specification that is displayed by default in the *Detailed Blockage Layers* field is determined by the technology file with which you are working. For example, if you specify `(Metal1 (Metal1 or Oxide)) (Metal2 (Metal2 or Oxide))`, the detailed blockage is created on `Metal1` and `Metal2` layer for, `Metal1` or `Oxide` and `Metal2` or `Oxide` layer-geometry in the design. For more information about layer-geometry specification, refer to Geometry Specification.

Tip

When you type a value in a field in the *Blockages* group box, the checker running in the background validates the specified value. If the specified value is invalid, a pink highlight appears around the text box. A warning message is also displayed in CIW explaining why the value is invalid.

Loading and Saving an options File

The options file is an editable text file that contains the abstraction rules used by Dynamic Abstract Generation to create abstracts. You can use the following methods to load and save an options file:

- Save the abstraction rules specified in the Abstract Pcells form to an options file.To save the abstraction rules to an options file, click *Save*. Specify the name of the options file to which you want to save the abstraction rules in the *options File Name* field. You can also load an existing options file, edit it, and save the changes.

  Note: By default, the name that appears in the *options File Name* field is *.dynamicAbstract.options*.

- Load the various rules contained in an options file to the Abstract Pcells form. To load the abstraction rules contained in an options file, select the required options file and click *Load*.

© 2020 All Rights Reserved.

- Directly edit an existing options file in a text editor and save the changes.

The following figure shows a sample options file.

dynamicAbstract.option (/net/cicsol7/export/home/parula/USV) - GVIM
File Edit Tools Syntax Buffers Window Help
;;;

Specifying Advanced options

To specify the advanced abstraction rules, click the *Advanced options* button. The Save Dynamic Abstract Generation options form is displayed as shown in the following figure.

Note: When you point to an option in the Abstract Pcells form, an info balloon that describes the purpose of the option is displayed

The form has three tabs.

- General tab
- Extraction tab

© 2020 All Rights Reserved.

- Antenna tab

The various advanced abstraction rules that are available on the three tabs are discussed in detail below.

© 2020 All Rights Reserved.

General Tab

You can specify the following settings on the General tab.

- Specifying Pin Settings
- Specifying prBoundary
- Specifying Blockages

Specifying Pin Settings

- Mapping text labels to pins

  In the *Map Text Labels to Pins* field, you can specify the text labels that you want to map from the layout view to be created as pins in the layout dual view. The *Map Text Labels to Pins* field is case-sensitive. The syntax for specifying the text labels is as follows:

  ```
  (textLPP1 geomLPP1 geomLPP2)
  (textLPP2 geomLPP3 geomLPP4)
  ```

  where,

  - The syntax of `textLPP` is `(textLayer textPurpose)` or `textLayer`.
  - The syntax of `geomLPP` is `(geomLayer geomPurpose)` or `geomLayer`.

  An example of this is as follows:

  ```
  (textLayer (metal1 drawing) (metal2 drawing))
  ```

  In this example, the text labels on layer `textLayer` for all purposes are mapped first to the shapes on layer `metal1` with purpose `drawing`, and then to shapes on layer purpose pair `metal2 drawing`.

  Note: Instead of specifying a layer purpose pair, if you specify only the layer name, ‘all’ purposes for the specified layer are considered during Dynamic Abstract Generation.

  For more information, refer to Mapping Text Labels to Pins.

- Specifying Pin Names

  In the *Power Pin Names*, *Ground Pin Names*, *Clock Pin Names*, *Analog Pin Names,* and *Output Pin Names* fields, specify a list of names for power pin, ground pin, clock pin, analog pin, and output pin, respectively. Specifying the pin names enables you to distinguish the pin types in the design. You can specify a list of pin names in the form of regular expressions, with each pin name separated by a space. For more information about regular expressions, refer to Writing Regular Expressions.

© 2020 All Rights Reserved.

By default, the *Clock Pin Names*, *Analog Pin Names*, and *Output Pin Names* fields are empty.

- Specifying Geometry Search Depth

  In the *Geometry Search Depth* box, specify the hierarchy depth — the number of levels in the design hierarchy the Dynamic Abstract Generation process must search for metal shapes that lie underneath a text label. The default value is `20`. For more information, refer to Specifying the Search Depth.

- Preserving Text Labels

  Select the *Preserve text labels* check box to retain the text labels from the layout view in the layout dual view.

Specifying prBoundary

In the *Using geometry on layers* field, specify a list of layers or layer purpose pairs to be used to define the prBoundary. The boundary is drawn so that it encloses all the geometries found on the specified layers or layer purpose pairs. The syntax for specifying the layers is as follows:

```
LPP1 LPP2
```

An example of this is as follows.

```
metal1 (metal2 drawing)
```

In this example, all geometries on layer `metal1` for all purposes and on layer `metal2` for `drawing` purpose are enclosed by prBoundary.

Specifying Blockages

In the Blockage Spacing Constraint field, specify a list of layer purpose pairs on which you want to define a spacing constraint along with the spacing value. For example, you can specify (METAL1 0.75) (METAL2 0.3). If you do not specify the value for the spacing constraint, than the default spacing value is the minSpacing value that is defined in the technology file for that particular layer. You can also specify the blockage spacing constraint in the Dynamic Abstract Generation options file as follows.

```
absSetOption( "AbstractBlockageUserDefinedSpacing" "(MET1 0.75) (MET2 0.3)")
```


© 2020 All Rights Reserved.

Extraction Tab

On the *Extraction* tab, you can specify the extraction information for signal and power nets.

Specifying Signal Extraction Information

In the *Signal Extraction* group box, you can specify the following options for extracting signal nets.

- Extract Signal Nets

  Select the *Extract Signal Nets* check box if you want to run connectivity extraction on signal nets. By default, the *Extract Signal Net* check box is deselected.

- Extract Layers

  In the *Extract Layers* field, specify the layers or layer purpose pairs on which signal net extraction is to be performed.

© 2020 All Rights Reserved.

- Weak Layers

  When extracting signal nets, you can set certain layers as weak layers and instruct the extractor to extract through these as well. In the *Weak Layers* field, specify the layers that you want to be considered as weak layers. The weakly connected pins will be created for the layers specified in the *Weak Layers* field. By default, the *Weak Layers* field is empty.

- Pin Layers

  In the *Pin Layers* field, specify the layers on which you want the pins to be created. This can be a subset of layers that are specified in the *Extract Layers* field.

- Extraction Depth

  In the *Extraction Depth* field, specify how many levels down the instance hierarchy the Dynamic Abstract Generation process must search for the shapes to be extracted. The default value is `32`, which is the maximum number of levels in a design hierarchy stored in the database.

- Extract Must Join Terms

  Select the *Extract Must Join Terms* check box to establish a must-join relationship between all disjoint groups of geometries extracted from separate pins on the same net.

- Must Join Term

  In the *Must Join Term* field, specify a list of terminal names, each separated by a space. If the extractor finds a terminal name that matches one of the specified terminal names, and if there are disjoint geometries from different pins on the net after extraction, a `MUSTJOIN` relationship is created for that terminal.

- Layer Connectivity

  In the *Layer connectivity* field, specify the connectivity between layers. This enables you to specify the layer connectivity without having to add via definitions in the technology file. The default value of this field is based on the connectivity derived from analyzing the vias in the technology file.

  The format for specifying a value in this field is to list the layers as shown below.

  ```
  (layer1 layer2 layer3) (layer4 layer5 layer6) ... (layer7 layer8) ...
  ```

  In the example above, a triple layer entry means that `layer1` connects to **`layer2`** through the via layer `layer3`. A pair means that the first layer, `layer7`, connects to the second layer, `layer8`, by overlap (for example, local interconnect to polysilicon).

  For more information, refer to Considerations for Specifying Layer Connectivity.

© 2020 All Rights Reserved.

Specifying Power Extraction Information

In the *Power Extraction* group box, you can specify the following options for extracting power nets.

- Extract Power Nets

  Select the *Extract Power Nets* check box to run connectivity extraction for power nets. If this check box is selected, all the pin shapes created during the Pins step are extracted. By default, the *Extract Power Nets* check box is deselected.

- Extract Layers

  In the *Extract Layers* field, specify the layers or layer purpose pairs through which you want the power nets to be extracted.

- Pin Layers

  In the *Pin Layers* field, specify the layers on which you want the pins to be created. Creating pins on a layer helps to determine the shapes found by the extractor that are converted into pins.

- Extraction Depth

  In the *Extraction Depth* field, specify how many levels down the instance hierarchy the Dynamic Abstract Generation process must search for the shapes to be extracted. The default value is `32`, which is the maximum number of levels in a design hierarchy stored in the database.

© 2020 All Rights Reserved.

Antenna Tab

By using the options on the *Antenna* tab, you can create process antenna data for Pcells. By default, all the check boxes on this tab are deselected.

- Calculating Antenna Data

  Dynamic Abstract Generation computes the process antenna data for a given layer by using the following options on the *Antenna* tab.

  - Hierarchical Antenna
  - Antenna for Input Pin
  - Antenna for Output Pin
  - Antenna for In-Out Pin
  - Extract Metal Area
  - Extract Metal Side Area

© 2020 All Rights Reserved.

For more information, refer to Performing Process Antenna Calculations.

- Selecting Diffusion Layers

  Select the *Diffusion Layers* check box to extract diffusion layers specified in the technology file.

- Specifying Antenna Layers

  In the *Antenna Layers* field, specify the layers for which you want to calculate their process antenna data.

- Defining Area

  In the *Define Gate Area* and *Define Drain Area* fields, specify the layer-geometry specification for defining the gate and drain geometry regions, respectively. An example that defines the gate area region is as follows:

  ```
  (poly and fiff)
  ```

  An example that defines the drain region area is as follows:

  ```
  (diff andnot poly)
  ```

  For more information, refer to Region.

- Define Oxide Number

  From the *Define Oxide Number* drop-down list, select the oxide model to be extracted for a particular diffusion layer. The list contains four oxide types: `Oxide1`, `Oxide2`, `Oxide3`, and `Oxide4`. You can apply a unique oxide model for up to four diffusion layers. For more information, refer to Oxide.

Unabstracting Pcells

Unabstracting a Pcell involves deleting the abstract data that exists for a Pcell. If an abstract data that you created for a Pcell does not meet your requirements and you want to create the abstract data again based on a new set of abstraction options, you need to first unabstract a Pcell and then run Dynamic Abstract Generation again.

© 2020 All Rights Reserved.

To unabstract a Pcell, in the Layout window, choose *Tools – Dynamic Abstract Generation For Pcells – UnAbstract Pcells*. The UnAbstract Pcells form is displayed. You can choose to unabstract all Pcells or the selected Pcells.

Highlighting Unabstracted Pcells

To highlight the Pcells that are marked for abstraction but have not yet been abstracted, in the Layout window, choose *Tools – Dynamic Abstract Generation From Pcells – Highlight Unabstracted Pcells*. A highlight is displayed around each Pcell that has not been abstracted, as shown below.

Note: If you click anywhere in the layout canvas, the highlights around the unabstracted Pcells are removed.

© 2020 All Rights Reserved.

Marking or Unmarking Pcells for Abstraction

To mark or unmark Pcells for abstraction, in the Layout window, choose *Tools – Dynamic Abstract Generation for Pcells – Mark/Unmark Pcells for Abstraction*. The Mark/Unmark Pcells for Abstraction form appears, as shown below.

You can either mark or unmark the Pcells for abstraction. To mark a Pcell for abstraction, select *Mark*. This is also the default option. To unmark a Pcell, select *Unmark*. As a result, the Pcell is ignored for the abstraction process. By default, all the Pcells are marked for abstraction.

Navigator Assistant (Using the Context-Sensitive Menu)

When the *Pcells* filter is selected from the *Show* drop-down list in the Navigator assistant, the *Abstract Status* column is displayed in the Navigator assistant. This column displays the dynamic abstract status for each Pcell by using an appropriate status icon. The following table describes the dynamic abstract status icons.

Icon	Name	Description
	Unmarked for abstraction	Indicates that the Pcell is unmarked for abstraction
	Dynamic Abstract does not exist (Marked for abstraction/ Unabstracted Pcell)	Indicates that the Pcell is marked for abstraction, but the abstraction data does not exist for the Pcell
	Dynamic Abstract exists (Abstracted Pcell)	Indicates that the abstraction data exists for the Pcell

You can right-click a Pcell in the Navigator Assistant and choose from the context-sensitive menu the required Dynamic Abstract Generation commands.

© 2020 All Rights Reserved.

Depending on the abstract status of a Pcell, the context-sensitive menu provides the *Dynamic Abstract* commands for Dynamic Abstract Generation. For example, if a Pcell displays the abstract status as “Dynamic Abstract does not exist”, the context-sensitive menu for the Pcell contains commands for unmarking the Pcell for abstraction, abstracting the Pcell, and unabstracting the Pcell, as shown in the figure below.

Note: Selecting the *Abstract Pcells* command from the context-sensitive menu does not display the Abstract Pcells form. The abstract is created automatically based on the abstraction rules specified in the options file.

© 2020 All Rights Reserved.

Layout Canvas (Using the Context-Sensitive Menu)

You need to select a Pcell in the layout canvas to view the context-sensitive menu. The Dynamic Abstract submenu that is available on the context-sensitive menu contains commands that can be used for Dynamic Abstract Generation. Right-click a selected Pcell to view this menu, as shown below.

Note: Selecting the *Abstract Pcells* command from the context-sensitive menu does not display the Abstract Pcells form. The abstract is created automatically based on the abstraction rules specified in the default options file, namely, `.dynamicAbstract.options`.

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

8

Multi-Patterning Technology Support in Abstract Generator (ICADVM20.1 Only)

This chapter discusses the following topics:

- Overview
- Prerequisites
- Color Representation for Pins
- Color Representation for Blockages
- Other Supported Blockage Colors
- Support for Trim Metal

Additional information:

- Multi-Patterning Technology (ICADVM20.1 Only)
- Enabling Coloring While Importing GDSII Information (ICADVM20.1 Only)

© 2020 All Rights Reserved.

Overview

Virtuoso® Abstract Generator has been enhanced to support Multi-Patterning Technology (MPT). To handle color, the abstract representation now includes the color of pins and obstructions.

The following diagram depicts the process of generating a color-annotated abstract cellview from a color-annotated layout cellview and exporting the abstract cellview to LEF 5.8.

© 2020 All Rights Reserved.

Pin and obstruction colors are represented in LEF 5.8 as:

```
      PORT
        LAYER MET1 ;
          POLYGON MASK1 1.108 0.432 0.352 0.432 0.352 0.4 1.076 0.4
1.076 0.112 0.344 0.112 0.344 0.08 1.108 0.08 ;
        LAYER MET0 ;
          RECT MASK1 1.038 0.084 1.062 0.166 ;
          RECT MASK1 1.038 0.346 1.062 0.428 ;
          RECT MASK1 0.87 0.084 0.894 0.166 ;
          RECT MASK1 0.87 0.346 0.894 0.428 ;
          RECT MASK1 0.702 0.084 0.726 0.166 ;
          RECT MASK1 0.702 0.346 0.726 0.428 ;
          RECT MASK1 0.534 0.084 0.558 0.166 ;
          RECT MASK1 0.534 0.346 0.558 0.428 ;
          RECT MASK1 0.366 0.084 0.39 0.166 ;
          RECT MASK1 0.366 0.346 0.39 0.428 ;
        LAYER VIA0 ;
          RECT 1.034 0.08 1.066 0.112 ;
          RECT 1.034 0.4 1.066 0.432 ;
          RECT 0.866 0.08 0.898 0.112 ;
          RECT 0.866 0.4 0.898 0.432 ;
          RECT 0.698 0.08 0.73 0.112 ;
          RECT 0.698 0.4 0.73 0.432 ;
          RECT 0.53 0.08 0.562 0.112 ;
          RECT 0.53 0.4 0.562 0.432 ;
          RECT 0.362 0.08 0.394 0.112 ;
          RECT 0.362 0.4 0.394 0.432 ;
      END
    END y
    PIN vdd
      DIRECTION INOUT ;
      USE POWER ;
      SHAPE ABUTMENT ;
      PORT
        LAYER MET2 ;
          RECT 0 0.496 1.176 0.528 ;
      END
    END vdd
    PIN vss
      DIRECTION INOUT ;
      USE GROUND ;
      SHAPE ABUTMENT ;
      PORT
        LAYER MET2 ;
          RECT 0 -0.016 1.176 0.016 ;
      END
    END vss
    OBS
      LAYER MET1 ;
        RECT MASK2 0.03 0.08 0.268 0.112 ;
        RECT MASK2 0.03 0.08 0.062 0.16 ;
        RECT MASK2 0.236 0.24 0.46 0.272 ;
        RECT MASK2 0.03 0.352 0.062 0.432 ;
        RECT MASK2 0.236 0.08 0.268 0.432 ;
        RECT MASK2 0.03 0.4 0.268 0.432 ;
        RECT MASK1 0.134 0.176 0.194 0.336 ;
        RECT MASK1 0.344 0.08 1.108 0.112 ;
        RECT MASK1 1.076 0.08 1.108 0.432 ;
```

Important

Abstracts do not include cluster information because routing applications such as Innovus may later shift the colors on a per-instance basis to avoid color conflict.

For more information about MPT, see *Virtuoso Multi-Patterning Technology User Guide*.

© 2020 All Rights Reserved.

Prerequisites

- The technology file must include the number mask.
- The following figure describes the licensing requirements for Abstract Generator when using Virtuoso Multi-Patterning Technology:

© 2020
All Rights Reserved.

Color Representation for Pins

This section provides information about the implementation of MPT in the following abstract generation steps.

- Pins Step
- Extract Step
- Abstract Step

Pins Step

Generating pins is the first step of the abstract generation process. Typically, in this step, Abstract Generator derives pin shapes from the text labels in the layout view and places them at the locations of the corresponding text labels.

To support MPT, in the layout editor, shapes are highlighted in different colors to represent different masks. During the Pins step, when searching for pin shapes and chasing shapes, Abstract Generator recognizes the effective color of shapes in the layout editor. Consequently, the pins that are generated inherit the colors of the shapes from which they are derived.

© 2020
All Rights Reserved.

The color lock attribute of pins is forward-propagated to the abstract view. Consider a color locked pin in the layout design. In the abstract view, the color lock attribute is applied to all the pins in the layer in which the pin is located.

For more information about the Pins step, see Pins Step.

For more information about MPT, see *Virtuoso Multi-Patterning Technology User Guide*.

© 2020 All Rights Reserved.

Extract Step

The Extract step is the second stage of abstract generation. This step traces the connectivity between shapes and terminals, starting at the pin shapes created in the Pins step.

To support MPT, during extraction, overlapping pins that lie on the same layer and on the same mask on the layout are merged in the abstract view. A single shape is generated by Abstract Generator.

Overlapping pins that are located on different masks on the layout are not merged in the abstract. Different pin shapes corresponding to the different masks are generated.

Situation 1:

The layout editor has two overlapping shapes on Mask 1 and a third shape is on Mask 2. Notice that during extraction, overlapping shapes on the same mask are merged, whereas a separate pin shape is generated for the pin that lies on a different mask.

© 2020
All Rights Reserved.

Situation 2:

In the layout editor, overlapping shapes are on different masks. During extraction, pin merging does not happen. Separate pin shapes for each pin are generated in the abstract.

For more information about the Extract step, see Extract Step.

For more information about MPT, see *Virtuoso Multi-Patterning Technology User Guide*.

Abstract Step

The Abstract step is the third step of the abstract generation process. In this step, you can use the options in the Adjust tab to adjust pin shapes for creating the final required pin shapes. These pin shapes are then fractured into rectangles.

MPT support has been extended to the options on the Adjust tab of the Abstract step. Options for performing the following tasks now recognize the colors of the original shapes in the layout editor:

- Creation of square boundary pins for signal and power nets
- Creation of ring pins for power nets
- Creation of follow ring pins for power nets
- Grouping the power geometries
- Setting the options for CLASS CORE ports
- Creating CLASS BUMP ports
- Specifying power rail widths, offsets, and shapes

© 2020 All Rights Reserved.

- Assigning layers for pin geometry
- Fracturing pins

  Pin shapes created using the above options inherit colors from the original shapes. For example, two pin shapes created by fracturing a single rectilinear shape will inherit the color of the original rectilinear shape.

  Consider two pin shapes, A and B located on mask 1 and mask 2, respectively. Pin A is rectangular, whereas pin B is polygonal in shape.

  During the abstract step, the rectangular pin shape is not fractured, whereas the polygonal pin shape is fractured into smaller rectangles. Thus, pin B is fractured into two rectangles, which remain on mask 2.

For more information about these options, see Abstract Step.

For more information about MPT, see *Virtuoso Multi-Patterning Technology User Guide*.

© 2020 All Rights Reserved.

Color Representation for Blockages

Similar to shapes, blockages can be assigned a specific or no color. Blockages with specific color can be interpreted the same way as shapes with specific colors. Blockages at the same level of hierarchy with a particular color in a layout view should be at same-color spacing distance from shapes of the same color.

Blockage colors depend on the type of blockage.

- Cover Blockage
- Shrink Blockage
- Detailed Blockage

Cover Blockage

A cover blockage covers an entire block for each metal layer used by the block. A cover blockage may have several underlying shapes of different colors. Therefore, cover blockages are assigned the ‘multi’ color option.

Shrink Blockage

A shrink blockage covers a set of shapes within a block. There can be multiple shrink blockages within a single block. Similar to cover blockages, shrink blockages can have

© 2020 All Rights Reserved.

multiple underlying shapes of various colors. Therefore, shrink blockages are also assigned the ‘multi’ color option.

Detailed Blockage

Detailed blockages cover individual geometries within blocks. This type of blockage is used for standard cells and possibly for higher-level routing layers in blocks. Unlike cover and shrink blockages, detailed blockage covers a single shape. Therefore, detailed blockages inherit the color of the parent shape.

© 2020
All Rights Reserved.

Other Supported Blockage Colors

Apart from the 'multi' and individual shape colors, Abstract Generator supports the following blockage colors.

- **Gray:** Pins and blockages corresponding to isolated shapes in the layout editor are assigned the gray color. Such pins are colorable, which means that their color is driven by their environment. Gray pins can be connected using any colored wire. During the Pins and Extract steps, if any gray metal shapes are coincident with any colored shape on the same layer, then the gray shape is dropped.
- **Black:** Blockages defined in the layout editor to keep routing out of a particular region are assigned the black color. Other colored shapes are kept at different-color spacing away from a black blockage.

During abstract generation, the region corresponding to a black blockage is kept clear of any shapes.

© 2020
All Rights Reserved.

Support for Trim Metal

Trim metal, also referred to as cut metal, is a physical layer used to trim portions of a metal layer to achieve tight end-to-end line spacing. Trim metals are defined in the technology file. They can be assigned to cut-colored (masks) or non-colored metal geometries.

A trim metal trims a specific color (mask) of metal in the layout. For example, `TM_1` trims `Metal2:mask1` and `TM_2` trims `Metal2:mask2`. Therefore, in situations where a trim metal cuts multiple metal layers, only a specific metal layer is trimmed.

Tip

Metal layers are cut during fabrication.

In the following diagrams, `TM_1` is a trim metal that cuts only `(Metal1 mask1)`; and `TM_2` cuts only `(Metal1 mask2)`. These trim metals:

- Do not have any effect on the same metal shapes drawn on other masks
- Do not affect any other metal

© 2020 All Rights Reserved.

In the first figure, there are continuous metal lines in the layout. These shapes are cut into two parts by trim metals, which can be connected to different nets, as shown in the second figure.

Trim metals can be assigned to:

- **Non–colored (gray) metal geometries:** You can specify the color of the metal layer to be trimmed.
- **Cut-Colored metal geometries:** Trims metal layers of the same color as the trim metal.

Abstract Generator has been enhanced to support trim metals. While loading a library, Abstract Generator automatically fetches information about trim metals from the technology file. Trim metals are recognized at each step of the abstract generation process. Therefore, at each step, metal layers are considered as effective after they are cut by trim metals in the previous step.

Note: Only metal shapes are cut by trim metals.

© 2020 All Rights Reserved.

The following table summarizes the effect of trim metals in each step of the abstract generation process:

Abstract Generation Step	Effect of Trim Metals
Pins	Pins under labels are cut.
Extract	Metal layers are cut during the following tasks: ■ Net extraction ■ Pin geometry restrictions ■ Antenna extraction
Abstract	■ Detailed blockages are cut. ■ Cover blockages are not cut.

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

A

Abstract Generator Menu Command Reference

This appendix describes the menu commands and the associated forms in the standalone Abstract Generator.

This chapter discusses the following topics:

- Form Features
- File Menu
- Bins Menu
- Cells Menu
- Flow Menu
- Help Menu

Note: Abstract Generator does not check the validity or correctness of the entered text, such as terminal name or layer name, against any syntax rules.

© 2020 All Rights Reserved.

Form Features

Feature	Description
OK button	Closes the form and puts into effect any changes made since opening it
Apply button	Puts into effect any changes made since the form was opened, or since the *Apply* button was last clicked
Cancel button	Closes the form, ignoring any changes made
	In forms with an *Apply* button, only those changes made since the last time *Apply* was clicked are ignored.
Help button	Displays Help information about the form
Text box	Allows for text entry
	Each text box displays a short help text when the cursor hovers over it.
Check box	Toggle an option on and off
Radio button	Specifies the value of a variable that takes one of several mutually exclusive values
Menu box	Specifies a criterion used to filter a list of values
Toolbar	Provides button shortcuts for frequently used commands

© 2020 All Rights Reserved.

File Menu

The *File* menu contains commands to perform the following tasks:

- Creating a Library
- Opening a Library
- Attaching Technology Information to a Library
- Importing GDSII Information
- Importing OASIS Data
- Importing LEF
- Importing DEF
- Importing Logical Data
- Importing Options File
- Exporting LEF
- Exporting a Report
- Exporting Options File
- Recording Commands and Events
- Replaying Commands
- Specifying Global Options
- Viewing the Statistics Report
- Closing Abstract Generator

© 2020 All Rights Reserved.

Creating a Library

The *File – Library – New* option allows you to create a new library in the specified directory. You can create a library at any time during the abstract generation process.

Selecting *File – Library – New* displays the *New Library* window.

The fields in this window are described below.

Specifying a Path for the Library Creation

In the *Library Path* field, specify a valid directory path to where the new library should be created. The default value is the current directory, that is, "`.`".

Click *Browse* to display the *Browse New Library Directory* form that can be used to locate the required directory location for the new library.

If no path is specified, a warning message is displayed requesting that you enter a valid directory path.

Specifying a Name for the New Library

in the *Library Name* field, specify a valid library name. The library name must not contain any spaces. You also cannot specify the name of an existing library that is already specified in the `cds.lib` file of the current directory.

After you click *OK*, a new library will be created at the location specified (a new library directory will be created if one does not already exist, with the same library directory name as the name of the library). The library is added to the `cds.lib` file and a copy of `cdsDefTechLib` is added to the library.

© 2020 All Rights Reserved.

Note: The default `cdsDefTechLib` technology library is copied into the new library to allow the import of technology LEF to create a technology file. When you are Importing LEF, you can chose to select the *Overwrite* option which will let you copy over the current technology.

© 2020 All Rights Reserved.

Opening a Library

You can specify the library you want to process in Abstract Generator by selecting *File – Library – Open* or by clicking the *Library* button in the toolbar.

Tip

If there is only one library registered, clicking the *Library* button on the toolbar automatically opens that library for the current session.

Selecting *File – Library – Open* displays the *Open Library* window.

The *Open Library* window lists the libraries and their associated paths that are currently registered in Abstract Generator. These are read in from the `cds.lib` file located in the run directory and are listed in an alphabetical order.

Note: The `cds.lib` file defines the libraries that tools read to identify the libraries they can use. The file maps user library names to physical directory paths. Usually, one `cds.lib` file, which might reference other `cds.lib` files, determines the libraries that are available to your design tool. For more information about `cds.lib`, see the chapter “Cadence Library Definition File” in the *Cadence Application Infrastructure User Guide*.

Important

The *File – Library – Open* menu-option is grayed out if there are no libraries listed in the `cds.lib` file.

Select the library you want to open and click *OK*. All the views created by Abstract Generator, including the final abstracts, are stored in this library.

The option settings from a previous Abstract Generator session are stored in your current working directory in a `.abstract.options` file. When you reopen the library, Abstract

© 2020 All Rights Reserved.

Generator reads in this and restores the bin options to the settings that existed when the library was last saved.

© 2020 All Rights Reserved.

Attaching Technology Information to a Library

The *File – Library – Attach Technology* option allows you to attach and associate existing technology library information to the current cell library.

Selecting *File – Library – Attach Technology* displays the *Attach Technology Library* form.

Important

The *Attach Library* menu option is available only if a cell library is already open.

This form lists all the libraries in the `cds.lib` file that have their own `techfile.cds` file contained in the library itself. The library list displayed also includes `cdsDefTechLib`, which is the default technology library.

Note: You cannot import a technology LEF file if the default technology file, `cdsDefTechLib`, is already attached to a cell library. If you attempt to do this, an error message will display.

If you choose to attach a library, Abstract Generator attempts to combine this library to the current library and load it into the software. If you attempt to attach the same technology file that is already associated with the current library, you will be given the opportunity to reattach that technology library.

© 2020 All Rights Reserved.

Importing GDSII Information

The *File – Import – Stream (GDSII)* option creates cellviews from existing layout information in GDSII format.

Important

The OpenAccess version of Abstract Generator uses XStream to import GDSII information.

Choose *File – Import – Stream (GDSII)* to start this command or click the *Layout* button in the toolbar. This displays the *Import Layout* form.

The fields of this form are explained below.

© 2020 All Rights Reserved.

Specifying GDSII Filenames and Their Mapping

In the *GDSII Filenames* field, specify the names of the GDSII files that you want to import. Use the *Browse* button to help you locate the files.

In the *Map GDSII names* cyclic field, you can specify whether you want GDSII names mapped to upper case, lower case, or to not be mapped at all.

Specifying the Layer Map File

A layer map file is an ASCII file that maps the layer numbers and data types used in the Stream file to the layer-purpose pairs in the OpenAccess database. For more information about layer map files, see *Chapter 2 of the Design Data Translator's Reference*.

In the *Layer Map Table* field, specify the name of the layer map file you want to import. Use the *Browse* button to help you locate the file.

Specifying the Default Bin

In the *Default Bin* cyclic field, you can select the bin into which the cells are imported.

Note: This option applies only to any *new* library cells being imported. Cells imported previously are not moved.

During import, Abstract Generator creates layout views for the cells in the library. Some types of cells, such as sites and generated vias, do not need to be processed, so Abstract Generator moves them directly into the *Ignore* bin. A cell is placed in the *Default Bin* only if it meets the criteria below.

- There is a valid layout view.
- The cell is not a via.
- The cell is not a site.

After importing, if you select the *Default Bin* in the main window, you see the imported cells listed in the right-hand pane. The check mark in the *Layout* column indicates the presence of a layout view. Cells are also checked for valid layers.

Specifying the Template Files

In the *Template Filename* field, specify a stream template file to be imported into Abstract Generator. Use the *Browse* button to help you locate template files.

© 2020 All Rights Reserved.

If you do not select a template file, Abstract Generator creates a default file called `PcsStreamIn.template` in the `.abstract/import` directory. This file has default option settings of *No Mapping* and *Core* bin and can be used to create your own template file.

Note: Regardless of whether a template is supplied or not, the values of the *GDSII Filenames* and the *Map GDSII names* options are always the values specified in the form.

For more information about the contents of the template file and stream translation in general, see *Chapter 2 of the Design Data Translator's Reference*.

Specifying the Reference Technology Libraries

The subsection `techControl`, added to the technology file for ITDB (Incremental Technology Database) support, necessitates the specification of a set of ordered reference libraries to allow inheritance from these libraries.

Abstract Generator allows you to specify one or more valid reference technology libraries in the *Technology reference library name* field. The specified referenced technology libraries must be present in the `cds.lib` file and must be readable.

Enabling Coloring While Importing GDSII Information (ICADVM20.1 Only)

Use the `ImportGDSIIEnableColoring` to specify whether coloring needs to be enabled while importing GDSII data. By default, the option is set to `"true"`.

```
absSetOption("ImportGDSIIEnableColoring"  "true")
```

Important

For color layouts to be generated, in addition to setting the `ImportGDSIIEnableColoring` option to `"true"`, ensure that layer map file includes mask information.

© 2020 All Rights Reserved.

Importing OASIS Data

The *File – Import – Stream (Oasis)* option creates cellviews from existing layout information in OASIS format. Choose *File – Import – Stream (Oasis)* to start this command. This displays the *Import Oasis* form.

The fields of this form are explained below.

Specifying OASIS Filenames and Their Mapping

In the *Oasis Filenames* field, specify the names of the OASIS files that you want to import. Use the *Browse* button to help you locate the files.

In the *Map Oasis names* cyclic field, you can specify whether you want OASIS names mapped to uppercase, lowercase, or not to be mapped at all.

© 2020 All Rights Reserved.

Specifying the Layer Map File

A layer map file is an ASCII file that maps the layer numbers and data types used in the Stream file to the layer-purpose pairs in the OpenAccess database. For more information about layer map files, see *Chapter 2 of the Design Data Translator's Reference*.

In the *Layer Map Table* field, specify the name of the layer map file you want to import. Use the *Browse* button to help you locate the file.

Specifying the Default Bin

In the *Default Bin* cyclic field, you can select the bin into which the cells are imported.

Note: This option applies only to any *new* library cells being imported. Cells imported previously are not moved.

During import, Abstract Generator creates layout views for the cells in the library. Some types of cells, such as sites and generated vias, do not need to be processed, so Abstract Generator moves them directly into the *Ignore* bin. A cell is placed in the *Default Bin* only if it meets the criteria below.

- There is a valid layout view.
- The cell is not a via.
- The cell is not a site.

After importing, if you select the *Default Bin* in the main window, you see the imported cells listed in the right-hand pane. The check mark in the *Layout* column indicates the presence of a layout view. Cells are also checked for valid layers.

Specifying the Template Files

In the *Template Filename* field, specify a stream template file to be imported into Abstract Generator. Use the *Browse* button to help you locate template files.

If you do not select a template file, Abstract Generator creates a default file called `PcsStreamIn.template` in the `.abstract/import` directory. This file has default option settings of *No Mapping* and *Core* bin and can be used to create your own template file.

Note: Regardless of whether a template is supplied or not, the values of the *Oasis Filenames* and the *Map Oasis names* options are always the values specified in the form.

For more information about the contents of the template file and stream translation in general, see *Chapter 2 of the Design Data Translator's Reference*.

© 2020 All Rights Reserved.

Specifying the Reference Technology Libraries

The subsection `techControl`, added to the technology file for ITDB (Incremental Technology Database) support, necessitates the specification of a set of ordered reference libraries to allow inheritance from these libraries.

Abstract Generator allows you to specify one or more valid reference technology libraries in the *Technology reference library name* field. The specified referenced technology libraries must be present in the `cds.lib` file and must be readable.

© 2020 All Rights Reserved.

Importing LEF

The *File – Import – LEF* option creates cellviews from LEF data. This option is likely to be used when creating blocks from DEF. The LEF data for the leaf cells should be imported, followed by the DEF data for the associated block.

Selecting *File – Import – LEF* displays the *Import LEF* form.

The fields of this form are explained below.

Specifying the LEF Files

The *LEF Filenames* field specifies the LEF files to import. You can type the name manually in the field or *Browse* for existing files with the `.lef` extension in the current directory.

Specifying a Cellview for Importing the LEF Files

The *View* field specifies the cellview to which a LEF macro will be saved. The default view is `abstract` but this can be changed so that the LEF macro is saved to another view, such as the `layout` view.

© 2020 All Rights Reserved.

Specifying the Reference Technology Libraries

The subsection `techControl`, added to the technology file for incremental technology database support, necessitates the specification of a set of ordered reference libraries to allow inheritance from these libraries. For more information about incremental technology databases, see the *Virtuoso Technology Data User Guide*.

Abstract Generator allows you to specify one or more valid reference technology libraries in the *Technology reference library name* field. The specified referenced technology libraries must be present in the `cds.lib` file and must be readable.

Overwriting the Existing Technology File

Selecting the *Overwrite* option enables Abstract Generator to overwrite the current technology file if the LEF file to be imported contains one. This option is off by default.

All kinds of LEF files can be imported, ones with both technology and geometry information and also the ones with only technology or only geometry information.

Note: You cannot import a technology LEF file if the default technology file, `cdsDefTechLib`, is already attached to a cell library. If you attempt to do this, an error message will display.

© 2020 All Rights Reserved.

Defining the Library Name

In the *Library Name* field, specify the name of the LEF library that needs to be imported.

The library and its associated technology file are loaded into the current abstract session. At this point, the corresponding GDS file, if available, will also be read.

When specifying the *Library Name*:

- If a library is already loaded in the current abstract session, then the name of the currently open library is displayed in the *Library Name* field. Either retain the same library name or specify a different value.
- If you have created a new library, then the name of the newly created library is displayed in the *Library Name* field. Either retain the same library name or specify a different value.
- If no library is currently loaded in AG, and user wants to create new library using the Import LEF form, then it is mandatory to specify the library name along with the LEF file name. Else an error message is displayed and the Import LEF operation fails.
- When you click *OK*, Abstract Generator opens the specified library. So, if you specify a library name other than the currently loaded library, then the newly created library is loaded.

Alternatively use option `LEFLibraryName`.

© 2020 All Rights Reserved.

Importing DEF

The *File – Import – DEF* command imports a netlist in DEF format and updates the design with the information.

A DEF file contains the design-specific information of a circuit and is a representation of the design at any point during the layout process. The DEF file is in ASCII format.

DEF conveys logical design data to and physical design data from P&R tools. Logical design data can include internal connectivity (represented by a netlist), grouping information, timing parameters and physical constraints. Physical data includes placement locations and orientations, routing geometry data, and logical design changes for backannotation.

Selecting *File – Import – DEF* displays the *Import DEF* form.

Specifying the DEF Files

The DEF Specifies the DEF files to import. You can type in the name manually or *Browse* for existing files with the `.def` extension in the current directory.

Specifying the Default Bin

In the *Default Bin* cyclic field, select the bin to which DEF data is to be imported.

Cells will be placed in the *Default Bin* only if they conform to the following criteria:

- There is a valid layout view.
- The cell is not a via.
- The cell is not a site.

© 2020 All Rights Reserved.

If any of this criteria is not met, the cells will be placed in the *Ignore* bin.

Creating an Abstract from a DEF Block

Abstract Generator can be used to generate abstracts and process antenna information for routed blocks.

Step 1: Import

1. Use the *File – Import – LEF* option to import your LEF file.

   Note: When generating process antenna data for a routed block, you only need standard cell LEF, which has antenna information on the terminals.

2. Select *File – Import – DEF* and import your DEF file.

Step 2: Run Pins Step

Assuming that the DEF file contains pins for VSS/VDD, with no geometry, you should now do the following:

1. Run the Pins step.
2. In the Extract step (*Power* tab) leave the *Extract power nets* option off.
3. In the Abstract step (*Adjust* tab) select the style of power pins that you want to create, for example, boundary or pin.

This solution should work irrespective of the standard cells inside the block having feedthru or abutment pins and no matter what style of power pin you select for the block itself.

If the DEF file does not contain power pin statements for VSS/VDD, add these statements to the DEF file pins section, as shown below:

```
VSS + NET VSS + DIRECTION INOUT + USE GROUND
VDD + NET VDD + DIRECTION INOUT + USE POWER
```


© 2020 All Rights Reserved.

Step 3: Run Extract Step

When calculating the process antenna effect (PAE), it will be necessary to run `do_extraction`.

The recommended way of doing this is to leave the *Extract signal nets* option off, select all the appropriate antenna calculation buttons on the *Antenna* tab, and list the polysilicon, contact and diffusion layers in the *Layer Assignment for Extraction* table.

Note: The recommended antenna calculation to be performed here is *Calculate hierarchical antenna*, which does not require `DIFF`, `CONT`, or `POLY` layers in the extraction.

With this set-up, Abstract Generator will `do_extraction` to generate the PAE data but will not create any new pins. The pin geometry would essentially be the same as defined in the DEF pins section.

Step 4: Run Abstract Step

In the DEF flow, where each pin in the DEF file contained geometry, it is recommended to turn the *Create boundary pins* option off.

For *Power Nets*, if the power pin geometry was defined in the DEF file, it is likely that this is the desired pin geometry in the abstract. In this case, it is recommended that both *Create boundary pins* and *Create ring pins* options be turned off.

However, if the DEF file contains no geometric pin shape, you should select the *Create boundary pins* option for *Signal Nets* and *Create boundary pins* or *Create ring pins* option for *Power Nets*.

© 2020 All Rights Reserved.

Importing Logical Data

The *File – Import – Logical* command creates cellviews from existing logical information, typically represented in Liberty or Verilog® formats.

Select *File – Import – Logical* or click the *Logical* button in the toolbar. This displays the *Import Logical* form.

Specifying the Logical Data Files

In the *Logical Filenames* field, specify the names of one or more files you want to import, or use the *Browse* button to help locate the file and add it to the selection.

Specifying the File Type

You can filter the search by using the *Filetype* list. Choose the required file format, and then click *OK* to import the selected file.

Abstract Generator creates cells with logical views in the current library directory. If the cell already exists, Abstract Generator does not create a new cell but adds logical view information to the other views present for the cell.

If you import logical information after you have run the Pins step, Abstract Generator confirms that the layout and logical views match before extracting terminal properties from the logical view.

Cells that have only a logical view go to the *Ignore* bin by default.

© 2020 All Rights Reserved.

Important

You cannot import Verilog or Liberty information if you have set the global option to pick up logical information from a `schematic` or `symbol` view. For more information, see Specifying Global Options.

© 2020 All Rights Reserved.

Importing Options File

The *File – Import – Options* command allows you to load an options file saved in a previous session. You can save the settings of a session and load these settings back when you start a new session using the *Import* command.

Choose *File – Import – Options* to display the *Import Options* form.

In the *Options Filename* field, specify the path and name of the file to import, or use *Browse* to look for the file through various folders.

Click *OK* to import the options file from the specified location.

© 2020
All Rights Reserved.

Exporting LEF

The *File – Export – LEF* command generates LEF descriptions for cells in one or more bins.

Selecting *File – Export – LEF* displays the *Export LEF* form.

The fields of this form are explained below:

Specifying the LEF File

In the *LEF Filename* field, specify a path and name for the LEF file to export or use the *Browse* button to specify a location.

Specifying the LEF Data to be Exported

Use the *Export Geometry LEF Data* and *Export Tech LEF Data* options to choose the combination of geometry LEF and technology LEF to be exported. All the LEF information is written to the specified *LEF Filename*. The following examples show the variants of LEF data that can be exported from Abstract Generator:

© 2020 All Rights Reserved.

Geometry LEF data

```
PIN A
  DIRECTION INPUT ;
  PORT
  LAYER METAL1 ;
  RECT 0.715 1.845 1.105 2.490 ;
  END
END A
```

Tech LEF data

```
LAYER METAL1
  TYPE ROUTING ;
  DIRECTION HORIZONTAL ;
  PITCH 0.56 ;
  WIDTH 0.23 ;
  SPACING 0.23 ;
  RESISTANCE RPERSQ 0 ;
  CAPACITANCE CPERSQDIST 0 ;
  THICKNESS 0.1 ;
  ANTENNALENGTHFACTOR 0.00208 ;
  CURRENTDEN 0 ;
END METAL1
```

Specifying the Cells to be Exported

Use the *Cell List Filename* field to export LEF information for only the specified cells. The list of cells, whose LEF information is to be exported, is specified in a cell list file, with one cell name per line. Specify the name of this cell list file in the field or use *Browse* to specify the file's location. This is an optional field. For related messages, see ABS-16103 and ABS-16104.

If you specify a cell list file, the *Export LEF for Bin* option gets disabled.

Specifying the LEF Version

Using the *LEF Version* cyclic field, select the version of LEF that you want to export.

Specifying the Default Bin for the LEF Data to be Exported

In the *Export LEF for Bin* cyclic field, select the bins for which you want to export the LEF data. You can export LEF for all the cells in your library or on a bin-by-bin basis.

© 2020 All Rights Reserved.

Specifying the Bus and Divider Characters

Using the *Bus characters* and *Divider character* cyclic fields, select the characters you want to use to represent buses (`< >` or `[ ]`) and pseudohierarchy (`/` or `|`), respectively, in your LEF.

Click *OK* to export the LEF files. The resultant LEF file can be imported into the target P&R system.

Including DATABASE MICRONS in the Exported LEF

To set `DATABASE MICRONS` values in the LEF information exported from the CDB Abstract Generator, you must set the `gec3DBUPerUU` value in your CDB technology file. For more information about how to set the `DATABASE MICRONS` values, see Setting DATABASE MICRONS in the Exported LEF.

Note: In OpenAccess, the `DATABASE MICRONS` value is taken directly from the `DBUPerUU` value defined in the technology file.

© 2020 All Rights Reserved.

Exporting a Report

The report function outputs a summary of the cells in your library. It contains statistical information about the whole library, such as the number of cells by type and the number of cells with errors or warnings, and more detailed information about each of the individual cells.

Choose *File – Export – Report* to start this command. This displays the *Export Report* form.

In the *Report Filename* field, specify a path and name for the file to be exported, or use the *Browse* function to help you specify the location. Click *OK* to export the report to the specified file location.

The report file contains:

- General information about the tool version
- Date and time the report was generated
- Details of the number of errors and warnings generated during the current session
- Total number of cells by type
- Number of cells with errors or warnings
- Number of cells with valid logical views but no layout views

© 2020 All Rights Reserved.

Exporting Options File

The *File – Export – Options* command saves the export option settings for the current session.

Choose *File – Export – Options* to display the *Export Options* form.

In the *Options Filename* field, specify a path and name for the file, or use the *Browse* function to help you specify the location.

The options file records settings for both global and bin options as well as default settings for various other commands. You can use the options file to import these settings the next time you use Abstract Generator, or to restore them later in the current session after completing any experiments with alternative settings.

Click *OK* to export the options to the specified location.

© 2020 All Rights Reserved.

Recording Commands and Events

The *Record* option controls the behavior of the record file, `abstract.record`.

Choose *File – Record* to start this command. This displays the *Record* form.

Use the *Record* form to change the name of the record file, change the way the file is updated, or switch off recording altogether.

In the *Record File* field, specify the name of the record file. Use the *Browse* function to help you locate an existing file or change the current directory. If you do not change the name of the record file, it will be overwritten during the next Abstract Generator session.

Use the *Mode* pull-down list to select whether the commands of this session should overwrite or be appended to the record file or switch off the record function.

Note: The `.record` file will not acknowledge the selection of *File – Exit*.

By default, `abstract.record` is the name of the record file. If a file with that name already exists, the existing file is renamed to `abstract.record.`*`n`* where n is equal to 1, and the new record file is named `abstract.record`.

If the name of the record file is not changed in subsequent sessions, the `abstract.record` file is renamed to `abstract.record.`*`(n+1)`* to maintain the older versions of the record file. Therefore, the greater the numeric value in the record file name, the recent (latest) that file is.

For example, if the file name is not changed in three successive sessions, the following record files are created:

- `abstract.record` (Current record file)
- `abstract.record.2` (Next oldest)

© 2020 All Rights Reserved.

- `abstract.record.1` (Oldest)

Therefore, the `abstract.record` file is always the current record file. The other files with a numeric suffix are backups of previous record files.

© 2020 All Rights Reserved.

Replaying Commands

Every operation you perform during a session is recorded in a `.record` file. This allows you to rerun any session in batch mode and create user-defined macros with a minimum of manual intervention. This file can then be replayed in either GUI or non-GUI mode.

Choose *File – Replay* to start this command. This displays the *Replay* form.

Specify the name of the file to be replayed in the *Replay Files* field. Use the *Browse* function to help you locate it if required.

Click *OK* to replay the session described in the file.

The contents of a sample replay file are shown below.

```
absSkillMode()
absSetLibrary("amsPLL")
absSelectCellFrom("pllclk25" "pllclk25")
absSetBinOption("Block" "PinsTextPinMap" "(Metal1 Metal1)(Metal2
Metal2)(Metal3 Metal3)(Metal4 Metal4)")
absAbstract()
absSetOption("ExportLEFVersion" "5.5")
absExportLEF()
absExit()
```

Note: When replaying a session, the record function is automatically turned off.

© 2020 All Rights Reserved.

Specifying Global Options

The *File – General Options* command displays the *General Options* form that lets you view and change options that apply to the overall Abstract Generator. The options are split into different categories, accessed using the *Views* and *General* tabs in the *General Options* form.

General Tab

The *General* tab provides the options to set the default bin, LEF version, and the purpose name(s) to be ignored during the flow steps.

Constraint group

Specifies the name of the constraint group of the technology database. Abstract Generator works with the specified constraint group and derives the required information, such as the valid vias and valid routing layers, only from that constraint group. By default, the *Constraint group* field is empty. You can specify the constraint group name in the *Constraint group* field. If you have specified a constraint group name, then Abstract Generator will search for the rules in the specified constraint group. If the rules are not found in the specified constraint group, Abstract Generator displays an error. However, if constraint group name is not specified, Abstract Generator will search for the rules in the constraint group of type `LefDefaultRouteSpec`. If the constraint group of type `LefDefaultRouteSpec` is not

© 2020 All Rights Reserved.

found, then Abstract Generator will search for the rules in the constraint group named `LefDefaultRouteSpec`. If the rules are not found in either the constraint group of type `LefDefaultRouteSpec` or in the constraint group named `LefDefaultRouteSpec`, Abstract Generator displays an error.

Pin purpose name

Sets the purpose for pins on the final abstracts. The default purpose is `net`.

Default Bin

Specifies the default bin into which library cells are imported.

Note: This setting only applies to any *new* library cells that are being imported. Any cells that have been imported previously will remain in their original import location.

Cells will only be placed in the *Default Bin* if they conform to the following criteria:

- There is a valid layout view.
- The cell is not a via.
- The cell is not a site.

If any of these criteria are not met, the cells will be placed in the *Ignore* bin.

LEF Version

Sets the version of LEF to be used by Abstract Generator. You can select from `5.4`, `5.5`, `5.6`, and `5.7`. The default version of LEF used by Abstract Generator is `5.7` for mature nodes and `5.8` for advanced nodes.

Ignored Purpose(s)

Allows you to specify the purpose name(s) that you want to be ignored when a layer name without a qualifying purpose name is specified for the following options in the Pin, Extract, and Abstract steps:

- Pins step:
  - *Map* tab: *Map text labels to pins* field
  - *Boundary* tab: *Using geometry on layers* field

© 2020 All Rights Reserved.

- Extract step:
  - *Signal* tab: *Geometry Specification* column in the *Layer Assignment for Signal Extraction* table
  - *Power* tab: *Geometry Specification* column in the *Layer Assignment for Power Extraction* table
  - *Antenna* tab: *Geometry Specification* column in the *Layer Assignment for Antenna Extraction* table
- Abstract step:
  - *Adjust* tab: *Geometry Specification* column in the *Layer Assignment for Pin Geometry* table
  - *Blockage* tab: *Geometry Specification* column in the *Layer Assignment for Blockages* table
  - *Overlap* tab: *Using geometry on layers* field

Abstract Generator allows you to include the shapes on the ignored purpose(s) during the Pin, Extract, and Abstract steps if you explicitly mention the layer and the purpose names in the geometry specification at the appropriate locations in the flow steps.

By default, this option is prepopulated with the purpose name `boundary`, so that the `boundary` purpose is ignored by default in all the flow steps.

Abstract Generator allows you to include the shapes on the ignored purposes during the Pin, Extract, and Abstract steps if you explicitly mention the layer and the purpose names in the option value in the various steps. For example, if you specify the purpose to be ignored as "`boundary`", a geometry specification containing the layerName "`M1`" would include all shapes on layer "`M1`" and exclude those on purpose "`boundary`". To include the shapes on the ignored purpose "`boundary`", you will need to change the geometry specification to "`M1 or M1 boundary`".

Show Bubble Help

Displays bubble help for various GUI options. It is selected by default. You can turn it off if required.

Copy Routing Constraints from layout

Enables Abstract Generator to copy the mixed signal routing constraints from the layout view to the abstract view.

© 2020 All Rights Reserved.

Views Tab

Use this tab to specify the file names that Abstract Generator would use while creating layout, logical, and abstract database files. The *Logical file name* field can also be used to specify an existing logical view (for example, `schematic` or `symbol`) from which logical information is to be picked up.

Layout View Name

Specifies the view name for layout information.

For example, if you import GDSII layout data into Abstract Generator, this is the name used for the database files that are generated.

Logical View Name

Specifies the view name for logical information.

For example, if you import Verilog information into Abstract Generator, this is the name used for the database files that are generated.

© 2020 All Rights Reserved.

You can also use this field to specify the name of an existing view containing logical information (for example, schematic or symbol). Abstract Generator automatically picks up logical information from the specified file.

Note: If you do specify `schematic` or `symbol` in this field, you cannot then import Verilog data using the *Import Logical* form. The entry in this field must be compatible with the type of data that is subsequently imported.

Abstract View Name

Specifies the name used for the abstract views created by Abstract Generator.

© 2020 All Rights Reserved.

Viewing the Statistics Report

At any point during the Abstract Generator run, choose *File – Statistics Report* to see a report window that displays the abstract generation status and statistics until that point. A separate Statistics Report is displayed for each selected cell. The report provides the following information about the abstract generation run on each cell:

© 2020 All Rights Reserved.

The < and > buttons are useful when multiple cells are selected for abstract generation. Use these buttons to navigate between reports for the various cells. A new statistics report is generated for each Abstract Generator run.

To save a report:

1. Click *Save As*. The Save Statistics window is displayed.

2. Specify a different *Cell Stats Filename*. The default filename is `cellSummary.report`. To select an existing file, click *Browse* and select the required file.

   Note: If a file with the specified name already exists, then a message indicating this is displayed. You can choose to either replace the existing file or save the report with a different filename.

3. Click *OK*.

The statistics report is stored in the text format.

© 2020 All Rights Reserved.

Closing Abstract Generator

The *File – Exit* option closes Abstract Generator.

© 2020 All Rights Reserved.

Bins Menu

The *Bins* menu contains commands to perform the following tasks:

- Adding a User Bin
- Renaming a User Bin
- Deleting a User Bin
- Setting Bin Options

© 2020 All Rights Reserved.

Adding a User Bin

The *Bins – Add* option adds a new user bin to the list of existing bins.

In the *Add User Bin* form, type the name of the new bin in the *Enter bin name* field. The bin name should contain any spaces.

In the *Choose bin type* cyclic field, select a bin from the drop-down list. The bin type determines the default options settings for the new bin. Therefore, for example, if you create a new user bin of type *Core*, the user bin options will be identical to those currently set for the default *Core* bin.

When you click *OK*, Abstract Generator creates the new bin and displays it at the bottom of the list in the Bins pane.

© 2020 All Rights Reserved.

Renaming a User Bin

The *Bins – Rename* option changes the name of the selected user bin.

In the *Enter new bin name* field in the *Rename User Bin* form, type the new name for the selected user bin and click *OK* to accept the change.

Note: You cannot change the names of system bins.

© 2020 All Rights Reserved.

Deleting a User Bin

Select the bin you do not need and want to delete, and choose *Bins – Delete* and confirm the deletion.

If the bin you are deleting is not empty, you will be prompted (the *Delete User Bin* form will be displayed) to specify where to put its cells before the bin is deleted.

The name of the selected user bin will display in the *Delete bin* field. In the *Move its cell to* list box, select the bin to which you want the cells of the deleted bin to be moved and click *OK*. The cells are moved and the user bin is deleted.

© 2020 All Rights Reserved.

Setting Bin Options

The *Bin Options* form allows you to view and change the options associated with system and user bins. These options control the way in which Abstract Generator processes the cells contained in any given bin.

© 2020 All Rights Reserved.

Choose *Bins – Options* to display the *Bin Options* form.

© 2020 All Rights Reserved.

When you run flow steps on selected cells, you see a modified version of the *Bin Options* form, which lets you view and change options relevant to the steps about to be run.

The *Bin Options* form comprises the following:

Bin Selector

The option settings are applied to the bin specified in the panel on the left of the form.

If there is a bin selected when you open the form, it will be selected here with the option settings for that bin displayed. If there are multiple bins selected, the first bin in the list is selected. You can view and change the options associated with any other bin by clicking the button next to the bin name.

Step Selector

The abstract generation flow is split into four steps: Pins, Extract, and Abstract. You access the options associated with each step using the step selector in the top left of the *Bins Options* form.

When you open the *Bin Options* form, the default selection is Pins. Use the buttons to view and modify options associated with the other steps.

© 2020 All Rights Reserved.

Tabs

The options associated with each flow step are further grouped into logical categories, accessed by clicking the appropriate tab at the top.

Input Fields

When setting options, consider the following rules:

Measurements	All options that take measurements, for example for coordinates or distances, are specified in µ.
Layers	Where an option takes a layer name, the name supplied must be defined in the technology file; otherwise, the entry is rejected and an error is generated when you try to move from the field in question.
Purpose	Only purposes defined in the technology file are valid.

Buttons

In addition to the *OK*, *Cancel*, and *Help* buttons that are commonly present on other Abstract Generator forms, the *Bins Options* form has the *Copy From Bin* button.

Use the *Copy From Bin* button to copy option settings from another bin to the selected bin. This is useful when creating a new user bin; you can selectively copy bin specific options only while copying options from one bin to another.

© 2020 All Rights Reserved.

Clicking the *Copy From Bin* button displays the *Copy Bin Options* form.

In the *From* field, select the bin whose options you want to copy and click *OK*. For example, the settings for the *Core* bin could be copied to *UserBin*. The current bin is entered automatically in the *To* field.

This function overwrites all the current option settings for the target bin.

The following table lists the options that are disabled for specific bins and are to be excluded from that bin. This means that the options listed under a particular bin will be excluded while copying options from other bin to this bin and vice-versa.

Bin Name	Excluded Options
I/O	AbstractAdjustBoundaryPinsSig
	AbstractAdjustBoundaryPinsPwr
	AbstractAdjustRingPinsPwr
	AbstractAdjustFollowRingPin
	AbstractSiteArrayPattern
	AbstractSiteArrayCells

© 2020 All Rights Reserved.

Core	AbstractAdjustBoundaryPinsSig
	AbstractAdjustBoundaryPinsPwr
	AbstractAdjustEdgeTowardsCore
	AbstractAdjustPowerGeometryGroupsEastWest
	AbstractAdjustClassCoreNets
	AbstractAdjustIncludeAllShapes
	AbstractAdjustCopyClassCorePort
	AbstractAdjustPinsTouchBoundary
	AbstractAdjustPinsTouchNonCoreEdge
	AbstractAdjustCreateClassBumpPort
	AbstractAdjustRingPinsPwr
	AbstractAdjustFollowRingPin
Block	AbstractAdjustEdgeTowardsCore
	AbstractAdjustPowerGeometryGroupsEastWest
	AbstractAdjustClassCoreNets
	AbstractAdjustIncludeAllShapes
	AbstractAdjustCopyClassCorePort
	AbstractAdjustPinsTouchBoundary
	AbstractAdjustPinsTouchNonCoreEdge
	AbstractAdjustCreateClassBumpPort
	AbstractSiteName
	AbstractSiteNameDefine
	AbstractSiteArrayPattern
	AbstractSiteArrayCells
	AbstractSiteArrayCellsTable

© 2020 All Rights Reserved.

Corner

AbstractAdjustBoundaryPinsSig

AbstractAdjustBoundaryPinsPwr

AbstractAdjustEdgeTowardsCore

AbstractAdjustPowerGeometryGroupsEastWest

AbstractAdjustClassCoreNets

AbstractAdjustIncludeAllShapes

AbstractAdjustCopyClassCorePort

AbstractAdjustPinsTouchBoundary

AbstractAdjustPinsTouchNonCoreEdge

AbstractAdjustCreateClassBumpPort

AbstractAdjustRingPinsPwr

AbstractAdjustFollowRingPin

AbstractSiteArrayPattern

AbstractSiteArrayCells

AbstractSiteArrayCellsTable

© 2020 All Rights Reserved.

Cells Menu

The *Cells* menu contains commands to perform the following tasks:

- Selecting Cells Based on a Criteria
- Selecting All Cells
- Deselecting All Cells
- Selecting Cells with Invalid Views
- Sorting Cells
- Distributing Cells into Bins
- Moving Cells Across Bins
- Viewing and Editing Cell Properties
- Viewing and Editing Cell Terminal Properties
- Viewing Cell Status Report
- Editing Cellviews in Virtuoso Layout Editor

© 2020 All Rights Reserved.

Selecting Cells Based on a Criteria

The *Cells – Select* command selects one or more cells using a variety of selection criteria.

Choose *Cells – Select* to start this command. This displays the *Select Cells* form.

© 2020 All Rights Reserved.

Using this form, you can specify criteria based on any of the following parameters:

Name	Enter a regular expression that is included in the cell names you want to select. All matching cells are selected, irrespective of the *View* set.
View	Select the view you are interested in. Only cells that have the specified *View* present and match any subsequent *Height*, *Property* or *Status* criteria are selected.
Height	Specify minimum or maximum cell heights, or a range within which the cell height must fall.
Property	Enter a system or user-defined cell property and a value for the cells you want to select.
Select By Error Messages	Select one or more of the error messages listed in the table to select cells in a library. Only cells that have had a specific, selected error message applied to them will be filtered through in the selection procedure. The figure shown above displays a list of error messages based on which you can select cells.

Click *OK* to select the cells matching the criteria and close the form, or use the *Apply* button to perform incremental selection without closing the form.

Note: The criteria specified are applied only to the cells in the current bin.

© 2020 All Rights Reserved.

Selecting All Cells

The *Cells – Select All* command selects all cells displayed in the *Cell* pane. Clicking the table header of the *Cell* pane also selects all the cells.

Note: This function does not select cells in bins that are not selected.

© 2020 All Rights Reserved.

Deselecting All Cells

When you click *Cells – Deselect All*, no cell in the cells pane is selected.

© 2020 All Rights Reserved.

Selecting Cells with Invalid Views

The *Cells – Revalidate Selected* command revalidates all invalid views for the cells selected in the *Cell* pane.

Abstract Generator invalidates a view if the view is out of date. It then turns the symbol gray for that view in the *Cell* pane. Abstract Generator does not allow you to proceed with the abstraction flow until you run the step again to create that view.

A view might become invalid in any of the following situations:

- The technology file is changed after the view was created.
- The preceding view in the flow has a newer timestamp.
- Abstract options for the step have changed after the view was created.
- The cell is moved to another bin.

In these situations, Abstract Generator, by default, invalidates the view to enforce consistency between the techfile, changed cellviews, and options. This helps the inexperienced user. However, in some cases this is overly restrictive for experienced users. The *Revalidate Selected* option provides the override functionality to these experienced users.

Note: The result of selecting the *Revalidate Selected* command is that the view status is no longer grayed out even if the view is invalid. However, the errors associated with the view would still need to resolved prior to running any subsequent steps.

The need for using the *Revalidate Selected* option arises from the situation in which you might want to process the cells in a single bin with slightly different options. To do this without the *Revalidate Selected* option, you would need to create a user bin for each cell, move cells to the different user bins and then process the abstract for each bin. With the *Revalidate Selected* option, you can retain the cells in a single bin, modify the options for them, and process the abstract as required before revalidating views.

For information about the symbols denoting cell status, see Viewing Status Information.

© 2020 All Rights Reserved.

Sorting Cells

The *Cells – Sort* command sorts the selected cells according to a variety of sorting criteria. The specified criteria are applied only to the selected cells displayed in the *Cell* pane and not to the overall library.

Selecting *Cells – Sort* command displays the *Sort Cells* form.

The form options are described below.

Selecting the Criterion for Sorting

Select from *None, Name, Height, Status,* or *Selection*:

- *Name* sorts the selected cell list alphabetically by cell name.
- *Height* sorts the selected cell list according to the heights of the cells in the specified *View*.
- *Status* sorts the selected cells according to the status of the *View* specified, listing cells in a sequence: unknown, warning, error, and invalid.
- *Selection* moves all the selected cells to the top of the cell table.

Selecting a View for Sorting

Select from *Layout*, *Logical*, *Pins*, *Extract*, or *Abstract*.

The *Height* and *Status* methods consider only the view specified here. Cells that do not have the specified view are not sorted.

© 2020 All Rights Reserved.

Selecting the Order of Sorting

Select *Ascending* or *Descending*.

When you have specified your criteria, click *OK* to sort the cells.

© 2020 All Rights Reserved.

Distributing Cells into Bins

The *Cells – Distribute* command distributes cells among the system and user bins according to cell height, name or type.

Selecting *Cells – Distribute* displays the *Distribute Cells Among Bins* form.

The *Method* group options allow you to specify the criteria you want to use. Choose from *Height*, *Name* and *Type*. In the *Cells* group, select whether you want to distribute *All* the cells currently displayed in the *Cell* pane or limit the operation to the set of cells *Selected* when you opened the form.

© 2020 All Rights Reserved.

Distributing Cells Based on Height

Selecting the *Height* method changes the lower part of the form to display the following:

You can specify a series of height ranges in µ, and the bins to which cells falling within a particular range are to be moved.

To ensure that no cells are missed, only the lower bound of each range is editable. When you change the lower bound of one range, the upper bound of the next range down is adjusted automatically.

When you are satisfied with the height ranges, click *OK* to distribute the cells.

Distributing Cells Based on Name

Selecting the *Name* method changes the lower part of the form to display the following:

You can specify strings of characters and target bins for cell names containing those strings.

For example, the settings shown above would move all the cells with names that include the string *block* to the *Block* bin, those including the string *io* to the *IO* bin, and so on.

Note: Any cells not covered by specific criteria fall into the “other” category and are moved to the bin specified.

© 2020 All Rights Reserved.

Distributing Cells Based on Type

Selecting the *Type* method changes the lower part of the form to display the following:

Using this form, you can distribute the cells to other system and user bins. The *Type* method distribution is based on the `CellType` attribute of a layout cell on OpenAccess.

The first column represents the `CellType` attribute. By default, the `CellType` attributes in the first column are mapped to the corresponding system bins in the second column. The choices in the second column corresponding to a specific type are restricted to the user-defined and system bins of the same type. For example, if *myBin* is a user-defined bin of the type *Block*, *myBin* would be available for selection with the *Block* system bin in the second column. For cellviews with an unknown `CellType` (*Other*), all system and user-defined bins come up as options in the *Ignore* cyclic list.

© 2020 All Rights Reserved.

Moving Cells Across Bins

The *Cells – Move* command moves one or more selected cells to a different bin.

In the *Cell* pane, select the cells you want to move either manually or by using *Cells – Select*, then choose *Cells – Move*. This displays the *Move Selected Cells* form.

Choose the target bin and click *OK*. The selected cells are moved into the selected bin. The status for all the views remains unchanged.

© 2020 All Rights Reserved.

Viewing and Editing Cell Properties

Selecting *Cells – Cell Properties* displays the *Cell Properties* form that lets you view and change the properties on cells. Select *Cells – Cell Properties* after selecting one or more cells in the *Cell* pane.

The selected cells are listed in the *Cell Properties* form. Also listed is the bin in which the cell is stored and the current settings for the *prCellClass* and *symmetry* properties. You can select more than one cell by using the usual `Shift` and `Ctrl` key bindings.

Important

Abstract Generator is able to set properties only on valid views that it has created (Pins, Extract and Abstract). If a cell contains none of these views or if the views are invalid, the entries in the table are grayed out and you cannot change the properties for that cell.

To change a property of a cell, perform the following steps:

1. Select one or more cells in the list and use the *Change property* cyclic field to select the property you want to change.

   This changes the Allowed Values in the second cyclic field.

2. Select the new value for the selected cells.
3. Click *Apply* to update the form. This will not update the database.
4. Click *OK* to accept all your changes and close the form.

© 2020 All Rights Reserved.

Allowed Values

prCellClass

The allowed values for the *prCellClass* property depend on the bin in which the cell resides. The table below shows the allowed *prCellClass* entries and the corresponding LEF descriptions for each bin.

Bin	Value	LEF
Core	`core`	`CLASS CORE`
	`feedthru`	`CLASS CORE FEEDTHRU`
	`tieHigh`	`CLASS CORE TIEHIGH`
	`tieLow`	`CLASS CORE TIELOW`
	`spacer`	`CLASS CORE SPACER`
	`antennaCell`	`CLASS CORE ANTENNACELL`
	`preEndcap`	`CLASS ENDCAP PRE`
	`postEndcap`	`CLASS ENDCAP POST`
	`wellTap`	`CLASS CORE WELLTAP`
IO	`pad`	`CLASS PAD`
	`input`	`CLASS PAD INPUT`
	`output`	`CLASS PAD OUTPUT`
	`inOut`	`CLASS PAD INOUT`
	`power`	`CLASS PAD POWER`
	`spacer`	`CLASS PAD SPACER`
	`areaIO`	`CLASS PAD AREAIO`

© 2020 All Rights Reserved.

Bin	Value	LEF
Block	`block`	`CLASS BLOCK`
	`cover`	`CLASS COVER`
	`ring`	`CLASS RING`
	`coverBump`	`CLASS COVER BUMP`
	`blackBox`	`CLASS BLACKBOX`
	`soft`	`CLASS BLOCK SOFT`
Corner	`bottomLeftEndcap`	`CLASS ENDCAP BOTTOMLEFT`
	`bottomRightEndcap`	`CLASS ENDCAP BOTTOMRIGHT`
	`topLeftEndcap`	`CLASS ENDCAP TOPLEFT`
	`topRightEndcap`	`CLASS ENDCAP TOPRIGHT`

symmetry

Specifies the symmetry for the selected cells.

R0 | *X* | *Y* | *X Y*

Note: The symmetry set using this option overrides all other symmetry settings on the cell.

© 2020 All Rights Reserved.

Viewing and Editing Cell Terminal Properties

The *Cells – Terminal Properties* command displays the *Terminal Properties* form that lets you edit terminal properties for the cells in your library. Select *Cells – Terminal Properties* after selecting one or more cells in the *Cell* pane.

Abstract Generator assigns the properties to the terminals in the database. If the logical information already contains these properties on terminals, Abstract Generator displays them in the *Terminal Properties* form and propagates them through various Abstract Generator views.

This form works in the same way as the *Cell Properties* form. The table contains an entry for each terminal (*Terminal* column) defined for each of the selected cells (*Cell* column) selected in the *Cell* pane.

Important

Abstract Generator is able to set properties only on valid views that it has created (Pins, Extract and Abstract). If a cell contains none of these views or if the views are invalid, the entries in the table are grayed out and you cannot change the properties for the terminals in question.

To change a terminal property of a cell, perform the following steps:

1. Select one or more terminals from the list and use the *Change property* cyclic field to select the property you want to change.

   This changes the Allowed Values in the second cyclic field.

2. Select the new value for the selected cells.

© 2020 All Rights Reserved.

3. Click *Apply* to make the change. This updates the form but not the database.
4. Click *OK* to accept all your changes and close the form.

Note: Terminal property settings are saved in the view of the current step. Therefore, when a step is run again, the terminal properties will be restored to the original values set prior to any user amendments.

Allowed Values

Field	Allowed Values

direction	`input`
	`output`
	`inout`
	`jumper`
	`tristate`
	`unused`

termType	`signal`
	`clock`
	`analog`
	`power`
	`ground`

shape	`feedthru`
	`abutment`
	`ring`

netExpr	Specify a valid terminal net expression in the following format: `[@gnd:%:gnd!]` To remove a net expression, select the terminal, click the *netExpr* column header and clear the field at the top of the form. For information about creating and editing net expressions, see Chapter 2, “Understanding Connectivity and Naming Conventions” in the *Virtuoso Schematic Editor L User Guide*.

© 2020 All Rights Reserved.

suppSensitivity	Power terminal names. For more information about setting this property, see Support for Inherited Connections in Abstract Generator.
groundSensitivity	Ground terminal names. For more information about setting this property, see Support for Inherited Connections in Abstract Generator.

Support for Inherited Connections in Abstract Generator

Abstract Generator provides support to inherited connections for managing multiple power and voltage supplies in designs through the properties *netExpr*, *suppSensitivity*, and *groundSensitivity*.

To set the supply sensitivity for a signal pin, the cell that contains the signal pin must have more than one power terminal. Similarly, to set the ground sensitivity for a signal pin, the cell containing the signal pin must have more than one ground terminal.

The value of the *suppSensitivity* property is displayed in the view of the current step only when you set the value after running the step. For example, if you want to see the property in Pins view, you must set its value after running the Pins step. To get information about the supply sensitivity in a LEF file, you must set its value after running the Abstract step and then export the LEF file.

The *Terminal Properties* form displays all power and ground terminals of the cells selected in the *Cell* pane. You can select single or multiple signal pins on the same or different cells to set their supply or ground sensitivities. The property value will be updated on a signal pin only if all of the following criteria hold true:

- The cell with the signal pin has more than one power (for setting supply sensitivity) or ground (for setting ground sensitivity) terminal.
- The value to set is a power (for setting supply sensitivity) or ground (for setting ground sensitivity) terminal on the cell with the signal pin.

If either of these conditions is not met, Abstract Generator will issue an appropriate warning message when you click *Apply*.

© 2020 All Rights Reserved.

Example

Consider a situation in which `CellA` and `CellB` have the following power, ground, and signal terminals:

Cell	Terminal	termType
`CellA`	`VDD`	`power`
	`VSS`	`ground`
	`in1`	`signal`
	`in2`	`signal`
`CellB`	`VDD`	`power`
	`VSS`	`ground`
	`out1`	`signal`
	`out2`	`signal`
	`PWR`	`power`
	`GND`	`ground`

In this situation, Abstract Generator populates the values `VDD` and `PWR` for the *suppSensitivity* property and the values `VSS` and `GND` for the *groundSensitivity* property.

Notice that `CellA` contains only one power and one ground terminal. So, it is not feasible to update the supply or ground sensitivity for the signal pins on this cell. `CellB` contains more than one power and ground terminals. To set the supply sensitivity on a signal pin of this cell, you can choose one of `VDD` or `PWR`. Similarly, to set the ground sensitivity on a signal pin of this cell, you can choose one of `VSS` or `GND`.

© 2020 All Rights Reserved.

The following table summarizes the values that can be set for the *suppSensitivity* and *groundSensitivity* properties for the signal pins on `CellA` and `CellB`.

Cell	Terminal	termType	suppSensitivity	groundSensitivity
CellA	VDD	power		
	VSS	ground		
	in1	signal	X	X
	in2	signal	X	X
CellB	VDD	power		
	VSS	ground		
	out1	signal	VDD, PWR	VDD, GND
	out2	signal	VDD, PWR	VDD, GND
	PWR	power		
	GND	ground		

‘X’ indicates that values cannot be set for *suppSensitivity* and *groundSensitivity* properties.

© 2020 All Rights Reserved.

Viewing Cell Status Report

The *Cells – Report* command opens a report window to display the status of the selected cells.

The cell report provides detailed information about the status of a cell, with explanations of any messages generated for the different views present. Only views that have a valid status are reported.

If more than one cells is selected, the arrow buttons allow you to move back and forward through the selected set.

© 2020 All Rights Reserved.

Editing Cellviews in Virtuoso Layout Editor

You can select *Cells – Edit – Layout* to view and edit any cellview that exists for a selected cell. When you do this, Abstract Generator invokes Virtuoso® Layout Suite L Editor.

Note: Before editing a cellview, ensure that you have the correct VLS L license and that VLS is set in your path.

You can choose to view and edit the following cellviews: *Layout*, *Pins*, *Extract*, and *Abstract*. You cannot edit a selected cell that does not have an available view or the view is invalid. Moreover, if a view is set to be *read-only*, the Layout Editor will also only be available in read-only mode for that view.

© 2020 All Rights Reserved.

For more information about the use and editing capabilities of VLS L, see the *Virtuoso Layout Suite L User Guide*.

© 2020 All Rights Reserved.

Flow Menu

The *Flow* menu contains commands to perform the following tasks:

- Running the Pins Step
- Running the Extract Step
- Running the Abstract Step

© 2020 All Rights Reserved.

Running the Pins Step

The *Flow – Pins* command runs the Pins step for the selected cells.

Before the step is run, the Running Form is displayed. Use this form to view and modify the bin options relevant to the Pins step.

When you are satisfied with the options settings, click *Run* to run the step.

For detailed information about the Pins step, see Pins Step.

© 2020 All Rights Reserved.

Running the Extract Step

The *Flow – Extract* command runs as far the Extract step for the selected cells.

When you select the command, the Running Form is displayed. Use this form to view and modify the bin options relevant to the Extract step and any prior steps that must be run.

When you are satisfied with the options settings, click *Run* to run the step.

Abstract Generator processes the cells as far as the Extract step, creating an extracted view, along with any of the predecessor views required.

For detailed information about the Extract step, see Extract Step.

© 2020 All Rights Reserved.

Running the Abstract Step

The *Flow – Abstract* command runs as far as the Abstract step for the selected cells.

When you select the command, the Running Form is displayed. Use this form to view and modify the bin options relevant to the Abstract step and any prior steps that must be run.

When you are satisfied with the options settings, click *Run* to run the step.

Abstract Generator processes the selected cells as far as the Abstract step, creating an abstract view, along with any of the predecessor views required.

For detailed information about the Abstract step, see Abstract Step.

© 2020 All Rights Reserved.

Running Form

The *Running* form appears whenever you run a flow step. The form is a modified version of the *Bin Options* form but offers access only to those bins and options relevant to the step or steps about to be run.

The label in the title bar indicates which step is about to be run, such as *Running Step Pins* indicate that the Pins step would be executed. The Step Selector provides access only to the options associated with those steps. The Bin Selector lists only those bins in which the selected cells reside.

For more detailed information about individual options, see Setting Bin Options.

Processing Cells in Different States

When you press a toolbar button to perform one of the main flow steps, Abstract Generator checks the selected cells to ascertain which steps have to be run for each cell.

Example

Suppose you have two cells selected in the *Cell* pane, Cell1 and Cell2. Cell1 has run as far as the Pins step and Cell2 as far as the Extract step. If you now run the Abstract step for both the cells, the following steps also must be run:

- Extract and Abstract steps for Cell1
- Abstract step for Cell2

Consequently, the *Running* form will include all the options related to the Extract and Abstract steps.

Note: You can access options of a step that has already been run. However, if you change an option for a step that has already been run for a particular cell, that step and all downstream steps for that cell will become invalid.

In the example above, you can also access the *Pins* options, even if all the cells have valid pins views. However, changing one of the *Pins* options will render the existing pins and extract views for Cell2 invalid, and these steps will be rerun for that cell.

© 2020 All Rights Reserved.

Help Menu

The Help menu contains the following commands:

Help Topics

Lists the contents of the *Virtuoso Abstract Generator User Guide*.

Menu Commands

Provides the link to the Abstract Generator Menu Command Reference chapter in the *Virtuoso Abstract Generator User Guide*.

SKILL Commands

Provides the link to the Abstract Generator SKILL Functions chapter in the *Virtuoso Abstract Generator User Guide*.

Tutorial

Provides the ink to the Tutorial: Generate Abstracts by Using the Standalone Abstract Generator chapter in the *Virtuoso Abstract Generator User Guide.*

Error Messages

Provides the link to the Abstract Generator Error and Warning Messages chapter in the *Virtuoso Abstract Generator User Guide.*

About Abstract Generator

Displays the version number and copyright notice.

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

B

Abstract Generator SKILL Functions

This manual contains information about the SKILL functions used in Abstract Generator.

Tip

You can use the same functions in Tcl mode by typing `absTclMode` in the *Command Entry Field*. Note, however, that Tcl does not accept parentheses.

List of Abstract Generator SKILL Functions

Importing

absImportGDS

absImportOasis

absImportLogical

absImportLEF

absImportDEF

absImportCTLF

absImportVerilog

absImportOptions

Exporting

absExportLEF

absExportReport

absExportOptions

Library

© 2020 All Rights Reserved.

absGetLibrary

absSetLibrary

absAttachTechLib

Selecting

absSelect

absSelectAllBins

absSelectCells

absSelectBin

absSelectBinFrom

absSelectCell

absGetSelectedCells

absSelectCellFrom

Deselecting

absDeselectAllBins

absDeselectCells

absDeselectBin

absDeselectBinFrom

absDeselectCell

absDeselectCellFrom

absDeselectCellsInList

Bin Operations

absGetBins

absGetBinType

absGetSelectedBins

absMoveSelectedCellsToBin

absRenameBin

© 2020 All Rights Reserved.

- absNewBin
- absDeleteBin
- absDeleteBinMoveCellsTo
- absCopyBinOptions

Options

- absGetOption
- absSetOption

Bin Options

- absGetBinOption
- absSetBinOption

Flow Functions

- absPins
- absExtract
- absAbstract
- absVerify

Cell Properties

- absGetCellProp
- absSetCellProp

Terminal Properties

- absGetTerminalProp
- absSetTerminalProp

Miscellaneous

- absDisableUpdate
- absDistributeCells
- absEnableUpdate
- absExit

© 2020 All Rights Reserved.

absRevalidateSelectedCells

absSort

absVersion

iagCreateMenuItem

iagGenAbstract

© 2020 All Rights Reserved.

Abstract Generator SKILL Functions Overview

Record File

To find out more about SKILL functions, examine the `abstract.record` file, which contains the record of a session expressed in SKILL functions.

To find a particular SKILL function, use the graphical user interface to perform the action (for example, setting a bin option), then close the program and examine the entries in the `abstract.record` file.

Entering Functions

When entering SKILL functions in Abstract Generator, the processes of entering parameters and options are different.

- Set parameters directly on the function line; for example,

```
absSetLibrary "libname"
```

- Set options prior to calling the function, using either the `absSetOption` or `absSetBinOption` functions; for example,

```
absSetOption "ImportLogicalFiles" "logic1"
absSetOption "ImportLogicalType" "Verilog"
absImportLogical
```

Completing Functions

If you type in part of a function and press the tab key, Abstract Generator tries to complete the function. If it is unable to do so, it presents a list of possible function names that begin with the partial string you entered.

For example, because all of the SKILL API functions begin with the string `abs`, if you type in `a` and press *Tab*, you will see a list of ALL the function names. Function completion works only for functions beginning with `abs`.

In **Tcl mode only**, if you need to know the parameters of a function quickly, type the function name and press *Return*. If the function requires arguments, Abstract Generator lists them. If there are no required arguments, the function is executed.

For example,

```
abstract> absSetLibrary
```


© 2020 All Rights Reserved.

```
Wrong # args. absSetLibrary name
```

The `absSetLibrary` function requires the library name to be supplied.

© 2020 All Rights Reserved.

Importing

Use the following functions to import data into Abstract Generator.

- absImportGDS
- absImportOasis
- absImportLogical
- absImportLEF
- absImportDEF
- absImportCTLF
- absImportVerilog
- absImportOptions

© 2020 All Rights Reserved.

absImportGDS

```
absImportGDS(
     )
     => 0 / 1
```

Description

Creates cellviews from GDSII layout data.

Arguments

None

Value Returned

`0`	Returns `0` if an error is encountered.
`1`	Returns `1` if the function executes successfully.

Associated Options

[ImportGDSIIFiles *filenames*]
[ImportGDSIICaseMap { No Mapping | to Uppercase | to Lowercase }]
[ImportGDSIITemplateFile *filename*]
[DefaultBin { Core | IO | Corner | Block | Ignore }]

© 2020 All Rights Reserved.

Option Descriptions

`ImportGDSIIFiles` *`filenames`*

Specifies one or more GDSII stream file names that you want to import into Abstract Generator.

`ImportGDSIICaseMap { No Mapping | to Uppercase | to Lowercase }`

Maps the cell names in GDSII stream files to lower or upper case or specifies that cell names are not to be mapped.

`ImportGDSIITemplateFile` *`filename`*

Specify one stream `.template` filename at a time to be imported into Abstract Generator.

`DefaultBin { Core | IO | Corner | Block | Ignore }`

Selects the bin that you want library cells to be imported to. The default setting is `Core`.

Examples

```
absSetOption "ImportGDSIIFiles" "stream1"
=> 1
absSetOption "ImportGDSIICaseMap" "to Uppercase"
=> 1
absSetOption "ImportGDSIITemplateFile" "stream1.template"
=> 1
absSetOption "DefaultBin" "Core"
=> 1
absImportGDS
=> 1
```

Imports a GDSII file called `stream1`, maps the cell names to upper case, and moves them into the `Core` bin. The template file `stream1.template` is imported at the same time.

© 2020 All Rights Reserved.

absImportOasis

```
absImportOasis(
     )
     => 0 / 1
```

Description

Creates cellviews from OASIS data.

Arguments

None

Value Returned

0	Returns 0 if an error is encountered.
1	Returns 1 if the function executes successfully.

Associated Options

[ImportOasisFiles *OasisFilenames*]
[OasisImportTechRefLib *libName*]
[ImportOasisCaseMap { No Mapping | to Uppercase | to Lowercase }]
[ImportOasisTemplateFile *filename*]
[DefaultBin { Core | IO | Corner | Block | Ignore }]
[ImportOasisEnableColoring { true | false }]
[ImportOasisLayerMapFile *layerMapFileName*]

© 2020 All Rights Reserved.

Option Descriptions

`ImportOasisFiles` *`OasisFilenames`*

Specifies the OASIS files to import.

`OasisImportTechRefLib` *`libName`*

Specifies the reference technology library to be used while importing the OASIS files.

`ImportOasisCaseMap { No Mapping | to Uppercase | to Lowercase }`

Specifies the casing to be followed while mapping cell names in the OASIS files: uppercase or lowercase. Also specifies if the cell names must not be mapped.

`ImportOasisTemplateFile` *`filename`*

Specifies a file that has the `.template` extension to be used during import.

`DefaultBin { Core | IO | Corner | Block | Ignore }`

Specifies the bin in which the imported library cells must be placed. The default value is `Core`.

`ImportOasisEnableColoring { true | false }`

Specifies whether coloring must be enabled when importing the OASIS files.

`ImportOasisLayerMapFile` *`layerMapFileName`*

Specifies the layer map file to be used.

Example

```
absSetOption( "ImportOasisFiles" "file1" )
=> 1
absSetOption( "ImportOasisLayerMapFile" "layerMapfile1" )
=>1
absSetOption( "ImportOasisCaseMap" "to Uppercase" )
=> 1
absSetOption( "ImportOasisTemplateFile" "file1.template" )
=> 1
absSetOption( "DefaultBin" "Core" )
=> 1
absSetOption( "ImportOasisEnableColoring" "true" )
=>1
absImportOasis
=> 1
```


© 2020 All Rights Reserved.

Imports an OASIS file named `file1`, maps the cell names to upper case, and moves them into the `Core` bin. Coloring is enabled and a layer map file is specified. The template file `file1.template` is imported at the same time.

© 2020 All Rights Reserved.

absImportLogical

```
absImportLogical(
     )
     => 0 / 1
```

Description

Creates cellviews from existing logical information, typically represented in LIB or Verilog formats.

Arguments

None

Value Returned

`0`	Returns `0` if an error is encountered.
`1`	Returns `1` if the function executes successfully.

Associated Options

[`ImportLogicalFiles` *`filenames`*]
[`ImportLogicalType { LIB | Verilog }`]

Option Descriptions

`ImportLogicalFiles` *`filenames`*

Specifies the names of one or more logical files to import.

`ImportLogicalType { LIB | Verilog }`

Specifies the logical file type format, `LIB` or `Verilog`. The default setting is `LIB`.

© 2020 All Rights Reserved.

Examples

```
absSetOption( "ImportLogicalFiles" "logic1" )
=> 1
absSetOption( "ImportLogicalType" "Verilog" )
=> 1
absImportLogical
=> 1
```

Imports a Verilog logical file called `logic1`.

© 2020 All Rights Reserved.

absImportLEF

```
absImportLEF(
     )
     => 0 / 1
```

Description

Creates cellviews from LEF data.

Arguments

None

Value Returned

`0`	Returns `0` if an error is encountered.
`1`	Returns `1` if the function executes successfully.

Associated Options

[ImportLefFiles *filenames*]
[ImportLefView *viewname*]
[ImportTechLef { true | false }]

© 2020 All Rights Reserved.

Option Descriptions

`ImportLefFiles` *`filenames`*

Specifies the LEF files to import.

`ImportLefView` *`viewname`*

Specifies the cellview that a LEF macro will be saved to.

`ImportTechLef { true | false }`

Control whether LEF macros are overwritten or appended to when antenna LEF is imported after geometry LEF. The default setting is `false`.

Example

```
absSetOption( "ImportLefFiles" "myleffile1" )
=> 1
absSetOption( "ImportLefView" "abstract" )
=> 1
absSetOption( "ImportTechLef" "false" )
=> 1
absImportLEF
=> 1
```

Imports a LEF file called `myleffile1` and specifies that LEF macro information be saved to the `abstract` view without overwriting any existing LEF data.

© 2020 All Rights Reserved.

absImportDEF

```
absImportDEF(
     )
     => 0 / 1
```

Description

Creates cellviews from DEF data.

Arguments

None

Value Returned

`0`	Returns `0` if an error is encountered.
`1`	Returns `1` if the function executes successfully.

Associated Options

[`ImportDefFiles` *`filenames`*]
[`DefaultBin { Core | IO | Corner | Block | Ignore }`]

Option Descriptions

`ImportDefFiles` *`filenames`*

Specifies the name of the DEF files to be imported.

`DefaultBin { Core | IO | Corner | Block | Ignore }`

Selects the bin that you want library cells to be imported to. The default setting is `Core`.

© 2020 All Rights Reserved.

Example

```
absSetOption( "ImportDefFiles" "mydeffile1" )
=> 1
absSetOption( "DefaultBin" "Core" )
=> 1
absImportDEF
=> 1
```

Imports a DEF file called `mydeffile1`and moves the views into the `Core` bin.

© 2020 All Rights Reserved.

absImportCTLF

```
absImportCTLF(
     )
     => 0 / 1
```

Description

Creates cellviews from CTLF logical data.

Arguments

None

Value Returned

`0`	Returns `0` if an error is encountered.
`1`	Returns `1` if the function executes successfully.

Associated Options

[`ImportCTLFFiles` *`filenames`*]

Option Descriptions

`ImportCTLFFiles` *`filenames`*

Specifies the names of the CTLF files to be imported.

Example

```
absSetOption "ImportCTLFFiles" "myctlffile1"
=> 1
absImportCTLF
=> 1
```

Imports a CTLF file called `myctlffile1`.

© 2020 All Rights Reserved.

absImportVerilog

```
absImportVerilog(
     )
     => 0 / 1
```

Description

Creates cellviews from Verilog logical data.

Arguments

None

Value Returned

`0`	Returns `0` if an error is encountered.
`1`	Returns `1` if the function executes successfully.

Associated Options

[`ImportVerilogFiles` *`filenames`*]

Option Descriptions

`ImportVerilogFiles` *`filenames`*

Specifies the name of the CTLF file to be imported.

Example

```
absSetOption "ImportVerilogFiles" "myverilogfile1"
=> 1
absImportVerilog
=> 1
```

Imports a Verilog file called `myverilogfile1`.

© 2020 All Rights Reserved.

absImportOptions

```
absImportOptions(
     )
     => 0 / 1
```

Description

Loads a previous session's saved options file from the specified location.

Arguments

None

Value Returned

`0`	Returns `0` if an error is encountered.
`1`	Returns `1` if the function executes successfully.

Associated Options

[`ImportOptionsFile` *`filename`*]

Option Descriptions

`ImportOptionsFile` *`filename`*

Specifies the path and name of the options file to be loaded. The default is `abstract.options`.

Example

```
absSetOption "ImportOptionsFile" "abstract.options"
=> 1
absImportOptions
=> 1
```

Imports the options from the file called `abstract.options` saved in the current working directory.

© 2020 All Rights Reserved.

Exporting

Use the following functions to export data from Abstract Generator.

- absExportLEF
- absExportReport
- absExportOptions

© 2020 All Rights Reserved.

absExportLEF

```
absExportLEF(
     )
     => 0 / 1
```

Description

Generates LEF files for cells in one or more bins. All exported LEF data is contained in the LEF *filename*.

Arguments

None

Value Returned

0	Returns 0 if an error is encountered.
1	Returns 1 if the function executes successfully.

Associated Options

[ExportLEFFile *filename*]
[ExportGeometryLefData { true | false }]
[ExportTechLefData { true | false }]
[ExportLEFVersion { 5.3 | 5.4 | 5.5 | 5.6 | 5.7}]
[ExportLEFBin { All | Core | IO | Corner | Block | Ignore }]

© 2020 All Rights Reserved.

Option Descriptions

`ExportLEFFile` *`filename`*

Specifies the path to and name of the LEF file to export.

`ExportGeometryLefData { true | false }`

Controls whether geometry LEF is exported. The default setting is `true`.

`ExportTechLefData { true | false }`

Specifies whether technology LEF is exported. The default setting is `true`.

`ExportLEFVersion { 5.3 | 5.4 | 5.5 | 5.6 | 5.7}`

Specifies the version of LEF to export. The default setting value is `5.4`, if the `sedsm` is in your path.

`ExportLEFBin { All | Core | IO | Corner | Block | Ignore }`

Specifies the bins that contains the cells for which you want LEF generated. The default setting is `All`.

Example

```
absSetOption( "ExportLEFFile" "file1.lef" )
=> 1
absSetOption( "ExportLEFVersion" "5.5" )
=> 1
absSetOption( "ExportGeometryLefData" "true" )
=> 1
absSetOption( "ExportTechLefData" "true" )
=> 1
absSetOption( "ExportLEFBin" "All" )
=> 1
absExportLEF
=> 1
```

Exports a LEF 5.5 compatible file called `file1.lef` containing LEF geometry and technology information for all the cells contained in all the bins.

© 2020 All Rights Reserved.

absExportReport

```
absExportReport(
     )
     => 0 / 1
```

Description

Outputs a summary of the cells in your library to a report file.

Arguments

None

Value Returned

`0`	Returns `0` if an error is encountered.
`1`	Returns `1` if the function executes successfully.

Associated Option

[ExportReportFile *filename*]

Option Description

`ExportReportFile` *`filename`*

Specifies the path to and name of the report file. The default is `abstract.report`.

Example

```
absSetOption( "ExportReportFile" "file1.rep" )
=> 1
absExportReport
=> 1
```

Exports a report file called `file1.rep` in the current working directory.

© 2020 All Rights Reserved.

absExportOptions

```
absExportOptions(
     )
     => 0 / 1
```

Description

Save the options settings for the current session to a specified location.

Arguments

None

Value Returned

`0`	Returns `0` if an error is encountered.
`1`	Returns `1` if the function executes successfully.

Associated Options

[`ExportOptionsFile` *`filename`*]

Option Descriptions

`ExportOptionsFile` *`filename`*

Specifies the path to and name of the options file. The default is `abstract.options`.

Example

```
absSetOption( "ExportOptionsFile" "abstract.options" )
=> 1
absExportOptions
=> 1
```

Saves the options in a file called `abstract.options` in the current working directory.

© 2020 All Rights Reserved.

Library

Use the following functions to set and get library names and attach technology information.

- absGetLibrary
- absSetLibrary
- absAttachTechLib

© 2020 All Rights Reserved.

absGetLibrary

```
absGetLibrary(
     )
     => library_name
```

Description

Returns the name of the current library.

Arguments

None

Value Returned

library_name	Name of the library loaded in the current session of abstract generator.

© 2020 All Rights Reserved.

absSetLibrary

```
absSetLibrary(
     name
     )
     => t / nil
```

Description

Specifies the library that you want to process through Abstract Generator. This involves either creating a new library and setting into it or setting into an existing library.

Arguments

name	Specifies the name of the library to be processed. **Note:** If you specify a library name that does not already exist, Abstract Generator creates a new library of that name.

Value Returned

`t`	Returns `t` on the successful execution of the function.
`nil`	Returns `nil` if an error is encountered.

Example

```
absSetLibrary( "MyLibrary" )
=> t
```

Sets the current library in Abstract Generator to be `MyLibrary`.

© 2020 All Rights Reserved.

absAttachTechLib

```
absAttachTechLib(
     name
     )
     => t / nil
```

Description

Attaches the named technology library to the current library.

Arguments

name	Specifies the name of the technology library to be attached. **Note:** You must set the current library before attaching a technology library.

Value Returned

`t`	Returns `t` on the successful execution of the function.
`nil`	Returns `nil` if an error is encountered.

Example

```
absSetLibrary( "MyLibrary" )
=> t
absAttachTechLib( "techlib1" )
=> t
```

Attaches the technology library, `techlib1`, to the current library, `MyLibrary`.

© 2020 All Rights Reserved.

Selecting

Use the following functions to select objects in Abstract Generator.

- absSelect
- absSelectAllBins
- absSelectCells
- absSelectBin
- absSelectBinFrom
- absSelectCell
- absGetSelectedCells
- absSelectCellFrom

© 2020 All Rights Reserved.

absSelect

```
absSelect(
     )
     => 0 / 1
```

Description

Selects cells according to the specified options.

Arguments

None

Value Returned

0	Returns 0 if an error is encountered.
1	Returns 1 if the function executes successfully.

Associated Options

[SelectInvalid { true | false }]
[SelectValid { true | false }]
[SelectPropertyVal *property value*]
[SelectPropertyName *property name*]
[SelectHeightTo *max height*]
[SelectHeightFrom *min height*]
[SelectView { All | Layout | Logical | Pins | Extract | Abstract | Verify }]
[SelectName *regular expression*]
[SelectError { true | false }]
[SelectByMsg *string*]

© 2020 All Rights Reserved.

Option Descriptions

`SelectInvalid { true | false }`

Specifies whether or not to select cells with an invalid status. The default setting is `false`.

`SelectValid { true | false }`

Specifies whether or not to select cells with a valid status. The default setting is `false`.

`SelectPropertyVal` *`property_value`*

Selects only cells that have a particular property value.

`SelectPropertyName` *`property_name`*

Selects only cells that have the named property as an attribute.

`SelectHeightTo` *`max_height`*

Selects only cells that do not exceed the specified height.

`SelectHeightFrom` *`min_height`*

Selects only cells that exceed the specified height.

`SelectView { All | Layout | Logical | Pins | Extract | Abstract | Verify }`

Selects only cells that have the specified view. The default setting is `All`.

`SelectName` *`regular_expression`*

Selects only cells with names matching the specified regular expression.

`SelectError { true | false }`

Specifies whether or not to select cells with status errors.

`SelectByMsg` *`string`*

Selects only cells with errors matching the specified string.

Example

```
absSetOption( "SelectValid" "true" )
=> 1
absSetOption( "SelectView" "Abstract" )
```


© 2020 All Rights Reserved.

```
=> 1
absSelect
=> 1
```

Selects all cells with a valid abstract view.

© 2020 All Rights Reserved.

absSelectAllBins

```
absSelectAllBins(
     )
     => integer
```

Description

Selects all the bins in the current library.

Arguments

None

Value Returned

integer	Returns the number of bins that Abstract Generator selects.

© 2020 All Rights Reserved.

absSelectCells

```
absSelectCells(
     )
     => integer
```

Description

Selects all cells in the selected bins.

Arguments

None

Value Returned

integer	Returns the number of cells that Abstract Generator selects.

© 2020 All Rights Reserved.

absSelectBin

```
absSelectBin(
     name
     )
     => 0 / 1
```

Description

Selects the named bin. This makes the specified bin current for use with other functions.

Arguments

name	Specifies the name of the bin to select.

Value Returned

0	Returns 0 if an error is encountered.
1	Returns 1 if the function executes successfully.

Example

```
absSelectBin( "Core" )
=> 1
```

Selects the Core bin and makes it the current bin.

© 2020 All Rights Reserved.

absSelectBinFrom

```
absSelectBinFrom(
     from
     to
     )
     => integer
```

Description

Selects a range of bins. The selection is made from top to bottom starting from the *from* bin. The *from* bin should be above the *to* bin the Bins pane, otherwise all bins listed after *from* are selected.

Arguments

from	Specifies the first bin to be selected in the range.
to	Specifies the last bin to be selected in the range.

Value Returned

integer	Returns the number of bins that Abstract Generator selects.

Example

```
absSelectBinFrom( "userbin1" userbin9" )
=> 2
```

Selects all bins in the range of `userbin1` to `userbin9` and returns the number of bins selected.

© 2020 All Rights Reserved.

absSelectCell

```
absSelectCell(
     name
     )
     => 0 / 1
```

Description

Selects a single named cell.

Arguments

name	Specifies the name of the cell to be selected.

Value Returned

0	Returns 0 if an error is encountered.
1	Returns 1 if the function executes successfully.

Example

```
absSelectCell "cell1"
=> 1
```

Selects cell1.

© 2020 All Rights Reserved.

absGetSelectedCells

```
absGetSelectedCells(
     )
     => string
```

Description

Returns the list of cells selected in the *Cell* pane. The cells might belong to different bins.

Arguments

None

Value Returned

string	Returns the list of selected cells.

Example

```
absGetSelectedCells
=> List of selected cells : cell1 BUF2 pll
```

Returns the currently selected cells in the *Cell* pane — `cell1`, `BUF2`, and `pll`.

© 2020 All Rights Reserved.

absSelectCellFrom

```
absSelectCellFrom(
     from
     to
     )
     => integer
```

Description

Selects a range of cells. The selection is made from top to bottom starting from the *from* cell. The *from* cell should be above the *to* cell the *Cell* pane, otherwise all cells listed after *from* are selected.

Arguments

from	Specifies the first cell to be selected in the range.
to	Specifies the last cell to be selected in the range.

Value Returned

integer	Returns the number of cells that Abstract Generator selects.

Example

```
absSelectCellFrom "ABC1" "ABC9"
=> 2
```

Selects all cells in the range from `ABC1` through `ABC9` and returns the number of cells selected.

© 2020 All Rights Reserved.

absSelectCellsInList

```
absSelectCellsInList(
     cellNameList
     )
     => integer
```

Description

Selects a list of cells that are not adjacent to each other in the *Cell* pane.

Arguments

cellNameList	Specifies a space-separated list of cell names to be selected.

Value Returned

integer	Returns the number of cells that Abstract Generator selects.

Example

```
absSelectCellsInList "cell1 cell2 cell6 cell9"
=> 4
```

Selects the cells `cell1`, `cell2`, `cell6`, and `cell9`, makes them current in Abstract Generator, and returns `4` as the number of the cells that Abstract Generator selects.

© 2020 All Rights Reserved.

Deselecting

Use the following functions to deselect objects in Abstract Generator.

- absDeselectAllBins
- absDeselectCells
- absDeselectBin
- absDeselectBinFrom
- absDeselectCell
- absDeselectCellFrom
- absDeselectCellsInList

© 2020 All Rights Reserved.

absDeselectAllBins

```
absDeselectAllBins(
     )
     => integer
```

Description

Deselects all currently selected bins.

Arguments

None

Value Returned

integer	Returns the number of cells that Abstract Generator selects.

© 2020 All Rights Reserved.

absDeselectCells

```
absDeselectCells(
     )
     => integer
```

Description

Deselects all cells in the current bins.

Arguments

None

Value Returned

integer	Returns the number of cells that Abstract Generator deselects.

© 2020 All Rights Reserved.

absDeselectBin

```
absDeselectBin(
     name
     )
     => 0 / 1
```

Description

Deselects a named bin.

Arguments

name	Specifies the name of the bin to deselect.

Value Returned

0	Returns 0 if an error is encountered.
1	Returns 1 if the function executes successfully.

Example

```
absDeselectBin "Core"
=> 1
```

Deselects the Core bin.

```
absDeselectBin "Core"
=> 0
```

The Core bin is already deselected.

© 2020 All Rights Reserved.

absDeselectBinFrom

```
absDeselectBinFrom(
     from
     to
     )
     => integer
```

Description

Deselects a range of bins. The deselection is done from top to bottom starting from the *from* bin. The *from* bin should be above the *to* bin the *Bin* pane, otherwise all bins listed after *from* are deselected.

Arguments

from	Specifies the first bin to be deselected in the range.
to	Specifies the last bin to be deselected in the range.

Value Returned

integer	Returns the number of bins that Abstract Generator deselects.

Example

```
absDeselectBinFrom "userbin1" "userbin9"
=> 3
```

Deselects all bins in the range of `userbin1` to `userbin9`.

© 2020 All Rights Reserved.

absDeselectCell

```
absDeselectCell(
     name
     )
     => 0 / 1
```

Description

Deselects a named cell.

Arguments

name	Specifies the name of the cell to be deselected.

Value Returned

0	Returns 0 if an error is encountered.
1	Returns 1 if the function executes successfully.

Example

```
absDeselectCell "cell1"
=> 1
```

Deselects `cell1`.

© 2020 All Rights Reserved.

absDeselectCellFrom

```
absDeselectCellFrom(
     from
     to
     )
     => integer
```

Description

Deselects a range of cells. The deselection is done from top to bottom starting from the *from* cell. The *from* cell should be above the *to* cell the *Cell* pane, otherwise all cells listed after *from* are deselected.

Arguments

from	Specifies the first cell to be deselected in the range.
to	Specifies the last cell to be deselected in the range.

Value Returned

integer	Returns the number of cells that Abstract Generator deselects.

Example

```
absDeselectCellFrom "ABC1" "ABC9"
=> 2
```

Deselects all cells in the range of `ABC1` to `ABC9`.

© 2020 All Rights Reserved.

absDeselectCellsInList

```
absDeselectCellsInList(
     cellNameList
     )
     => integer
```

Description

Deselects a list of cells that are not adjacent to each other in the *Cell* pane.

Arguments

cellNameList	Specifies a space-separated list of cell names to be deselected.

Value Returned

integer	Returns the number of cells that Abstract Generator deselects.

Example

```
absDeselectCellsInList "cell1 cell2 cell6 cell9"
=> 4
```

Deselects `cell1`, `cell2`, `cell6` and `cell9`.

© 2020 All Rights Reserved.

Bin Operations

Use the following functions to manipulate bins in Abstract Generator:

- absGetBins
- absGetBinType
- absGetSelectedBins
- absMoveSelectedCellsToBin
- absRenameBin
- absNewBin
- absDeleteBin
- absDeleteBinMoveCellsTo
- absCopyBinOptions

© 2020 All Rights Reserved.

absGetBins

```
absGetBins(
     )
     => string
```

Description

Displays the list of available bins. The list returned includes the five system bins as well as the user bins, if any.

Arguments

None

Value Returned

string	Returns the list of available bins.

Example

```
absGetBins
=> List of bins : Core IO Corner Block Ignore
```

Displays all the system bins.

```
absGetBins
=> List of bins : Core IO Corner Block Ignore myBin
```

Displays the five system bins and the user bin `myBin`.

© 2020 All Rights Reserved.

absGetBinType

```
absGetBinType(
     name
     )
     => string
```

Description

Returns the type of the bin passed as an argument. Both system and user bins can be specified as arguments.

The bin types of the system bins are specified below.

Core --> standard

IO --> IO

Corner --> corner

Block --> macro

Ignore --> none

Arguments

name Specifies the name of the bin whose type is to be determined.

Value Returned

string Returns the type of the specified bin.

© 2020 All Rights Reserved.

Example

```
absGetBinType "IO"
=> Bin type of bin 'IO' : IO
```

Displays the type of the bin `IO`.

```
absGetBinType "Block"
=> Bin type of bin 'Block' : macro
```

Displays the type of the `Block` bin.

```
absGetBinType "myBin"
=> Bin type of bin 'myBin' : corner
```

Displays the type of the user bin `myBin` as `corner`.

© 2020 All Rights Reserved.

absGetSelectedBins

```
absGetSelectedBins(
     )
     => string
```

Description

Returns the list of bins selected in the *Bin* pane. The order of the bins in the list matches the order in which they appear in the *Bin* pane.

Arguments

None

Value Returned

string	Returns the list of the selected bins.

Example

```
absGetSelectedBins
=> List of selected bins : Core
```

Returns `Core` as the bin selected in the *Bin* pane.

```
absGetSelectedBins
=> List of selected bins : IO Corner Ignore
```

Returns the list `IO Corner Ignore` as the bins selected in the *Bin* pane.

© 2020 All Rights Reserved.

absMoveSelectedCellsToBin

```
absMoveSelectedCellsToBin(
     name
     )
     => integer
```

Description

Moves any currently selected cells to a named bin. This can be one of the five system bins or a user-defined bin.

Arguments

name Specifies the name of the bin to which the selected cell(s) are to be moved.

Value Returned

integer Returns the number of cells moved.

Example

```
absMoveSelectedCellsToBin "IO"
=> 2
```

Moves the selected cells to the `IO` bin and returns the number of cells moved, which is two in this case.

© 2020 All Rights Reserved.

absRenameBin

```
absRenameBin(
     name
     new_name
     )
     => t / nil
```

Description

Renames a previously created user bin.

Note: You cannot rename a system bin.

Arguments

name	Specifies the name of the user bin to be renamed.
new_name	Specifies the new bin name. Do not enter any spaces in the name.

Value Returned

`t`	Returns `t` if the function executes successfully.
`nil`	Returns `nil` if an error is encountered.

Example

```
absRenameBin "userbin1" "userbin2"
=> t
```

Renames the bin `userbin1` to `userbin2`.

© 2020 All Rights Reserved.

absNewBin

```
absNewBin(
     name
     { Core | IO | Corner | Block | Ignore }
     )
     => t / nil
```

Description

Creates a new user bin to store library cells.

Arguments

name	Specifies the name of the new user bin to create. Do not enter spaces in the name.
`{ Core | IO | Corner | Block | Ignore }`	Specifies the user bin type.

Value Returned

`t`	Returns `t` if the function executes successfully.
`nil`	Returns `nil` if an error is encountered.

Example

```
absNewBin "userbin1" "Core"
=> t
```

Create a new user bin called `userbin1` which has the bin type `Core`.

© 2020 All Rights Reserved.

absDeleteBin

```
absDeleteBin(
     name
     )
     => t / nil
```

Description

Deletes the named user bin. You cannot delete a system bin.

Arguments

name	Specifies the name of a user bin to be deleted.

Value Returned

`t`	Returns `t` if the function executes successfully.
`nil`	Returns `nil` if an error is encountered.

Example

```
absDeleteBin "userbin1"
=> t
```

Deletes the bin `userbin1`.

© 2020 All Rights Reserved.

absDeleteBinMoveCellsTo

```
absDeleteBinMoveCellsTo(
     name
     { Core | IO | Corner | Block | Ignore | user_bin_name }
     )
     => t / nil
```

Description

Deletes a named user bin and moves any cells in that bin to another specified bin.

Arguments

name	Specifies the name of the user bin to be deleted.
`{ Core | IO | Corner | Block | Ignore |` *user_bin_name* `}`	Specifies the name of the bin into which any cells in the deleted bin are moved.

Value Returned

`t`	Returns `t` if the function executes successfully.
`nil`	Returns `nil` if an error is encountered.

Example

```
absDeleteBinMoveCellsTo "userbin1" "Core"
```

Deletes a user bin called `userbin1` and moves any cells previously contained in it to the `Core` bin.

© 2020 All Rights Reserved.

absCopyBinOptions

```
absCopyBinOptions(
     from_bin { Core | IO | Corner | Block | Ignore | user_bin_name }
     to_bin { Core | IO | Corner | Block | Ignore | user_bin_name }
     )
     => t / nil
```

Description

Copies bin option settings from one bin to another.

Arguments

from_bin	Specifies the name of the bin from which to copy the option settings. Valid values include `Core`, `IO`, `Corner`, `Block`, `Ignore`, and *`user_bin_name`*.
to_bin	Specifies the name of the bin to which the option settings are to be copied.

Value Returned

`t`	Returns `t` if the function executes successfully.
`nil`	Returns `nil` if an error is encountered.

Example

```
absCopyBinOptions "Core" "userbin1"
=> t
```

Copies the option settings from the `Core` bin to the user bin called `userbin1`.

© 2020 All Rights Reserved.

Options

Use the following functions to get and set global options in Abstract Generator.

- absGetOption
- absSetOption

© 2020 All Rights Reserved.

absGetOption

```
absGetOption(
     name
     )
     => string
```

Description

Returns the value of the specified global option.

Arguments

name Specifies the global option value to display. You can select from the following list:

- Import Options
- Export Options
- Library Options
- Select Options
- Deselect Options
- Distribute Options
- Sort Options

Value Returned

string Returns the value of the specified option.

Associated Options

[ExtractShapeLimit]
[ExcludePurposeList *purpose_names*]
[UseConstraintGroup *constraintgroup name*]

© 2020 All Rights Reserved.

Option Descriptions

`ExtractShapeLimit`

Returns the limit set for the number of shapes to be extracted in the given design. The default value is `10000`. To increase the limit, use this option with the `absSetOption` function.

`ExcludePurposeList` *`purpose_names`*

Returns the purpose name(s) to be ignored when a layer name without a qualifying purpose name is specified in the geometry specifications in the Pin, Extract, and Abstract steps.

`UseConstraintGroup` *`constraintgroup_name`*

Returns the name of the technology database constraint group that Abstract Generator uses to get information such as valid vias and valid routing layers.

Example

```
absGetOption "DefaultBin"
=> Core
```

Returns the current value set for the `DefaultBin` option.

© 2020 All Rights Reserved.

absSetOption

```
absSetOption(
     name
     value
     )
     => 0 / 1
```

Description

Sets values for all abstract generator global options. The SKILL function to apply the `absSetOption` settings is run after the `absSetOption` function has been set.

© 2020 All Rights Reserved.

Arguments

name	Specifies the option to be set. You can select from the following list: ❑ Import Options ❑ Export Options ❑ Library Options ❑ Select Options ❑ Deselect Options ❑ Distribute Options ❑ Sort Options
value	Specifies the option value to be used. For the list of valid values for each of the above listed options, refer to the respective option details.

Value Returned

0	Returns 0 if an error is encountered.
1	Returns 1 if the function executes successfully.

Associated Options

[ExtractShapeLimit *value*]
[ExcludePurposeList *purpose_names*]
[UseConstraintGroup *constraintgroup_name*]
[OptimizeBigDesignMemory]
[QuickAbstract]
[AnnotateLayoutDualView]
[ReadVoltageAndPurposeRules]
[CopyMSRoutingConstraints]
[ReadOnlyTechnology]
[numOfPointsInEllipseOrDonut]
[LEFLibraryName]

© 2020 All Rights Reserved.

Option Descriptions

`ExtractShapeLimit` *`value`*

Sets the limit for the number of shapes to be extracted in the given design. The default value is `10000`. To increase the limit, set the value as a positive number greater than `10000`.

`ExcludePurposeList` *`purpose_names`*

Specifies the purpose name(s) to be ignored when a layer name without a qualifying purpose name is specified in the geometry specifications in the Pin, Extract, and Abstract steps.

`UseConstraintGroup` *`constraintgroup_name`*

Specifies the name of the technology database constraint group that Abstract Generator uses to get information such as valid vias and valid routing layers.

`OptimizeBigDesignMemory`

Solves the memory issue which comes on the big block designs in Abstract Generator. By default, this option is set to False. To optimize the memory for big designs, set this option to True.

Note: Even when the OptimizeBigDesignMemory option is set to true and you still have memory issue with large design size on 32bit executable, suggest running the 64 bit executable.

`QuickAbstract`

Improves performance of Abstract Generator for small and mid size designs. If this option is set to true, Abstract Generator will not generate the intermediate cellviews, that is Pin and Extract cellviews. Only Abstract cellviews will be generated. This will result in 30% performance gain. By default, this option is set to false.

`AnnotateLayoutDualView`

© 2020 All Rights Reserved.

Enables Abstract Generator to annotate the abstract data in the layout itself and generate the layout dual view instead of the regular abstract view if set to `TRUE`. By default, this option is set to `FALSE`.

`ReadVoltageAndPurposeRules`

Enables Abstract Generator to read the layer-purpose-based and voltage dependent spacing rules from the technology file if set to `TRUE`. By default, this option is set to `FALSE`.

```
absSetOption(
"ReadVoltageAndPurposeRules" "true")
```

`CopyMSRoutingConstraints`

Enables Abstract Generator to copy the mixed signal routing constraints from the layout view to the abstract view, if set to `TRUE`. By default, this option is set to `FALSE`.

```
absSetOption("CopyMSRoutingConstraints"
"true")
```

`ReadOnlyTechnology`

Enables Abstract Generator to work with the read-only technology file. When this option is set to `TRUE`, Abstract Generator does not try to save to the technology file the layer purpose pairs that it creates. By default, this option is set to `FALSE`.

```
absSetOption("ReadOnlyTechnology" "true")
```

`numOfPointsInEllipseOrDonut`

Specifies the number of points that polygons should have before fracturing the blockages. These polygons represent elliptical, circular, or donut shapes in the layout. By default, this option is set to `64`.

```
absSetOption("numOfPointsInEllipseOrDonut
" "64")
```

`LEFLibraryName`

© 2020 All Rights Reserved.

Specifies the LEF library that needs to be imported and displayed in the current abstract session.The specified LEF library is opened in Abstract Generator after reading the corresponding technology LEF file. At this point, the corresponding GDS file, if available, will also be read.

```
absSetOption("LEFLibraryName" "t8")
```

`suppressMessage`

Suppress messages that have the specified message IDs.

```
absSetOption("suppressMessage" "ABS-
12005" "ABS-15009")
```

Example

```
absSetOption "DefaultBin" "Block"
=> 1
```

Sets the `Block` bin to be the default bin.

© 2020 All Rights Reserved.

Bin Options

Use the following functions to set and get bin options in Abstract Generator.

- absGetBinOption
- absSetBinOption

© 2020 All Rights Reserved.

absGetBinOption

```
absGetBinOption(
     bin
     option
     )
     => string
```

Description

Returns the value of a specified option in a given bin.

Arguments

bin	Specifies the name of the bin with which the option is associated. Valid values include `Core`, `IO`, `Corner`, `Block`, `Ignore`, and *`user_bin_name`*.
option	Specifies the option value to display. You can select from the following list: Pin Options Extract Options Abstract Options Verify Options

Value Returned

string	Returns the value of the specified option.

Example

```
absGetBinOption "Core" "PinsPowerNames"
=> ^((V(DD|CC))|(v(dd|cc)))(!)?$
```

Displays the name of the power pins in the `Core` bin.

© 2020 All Rights Reserved.

absSetBinOption

```
absSetBinOption(
     bin
     option
     value
     )
     => t / nil
```

Description

Sets a value for a specified option in a specified bin.

Arguments

bin	Specifies the name of the bin for which the option is to be set. Valid values include `Core`, `IO`, `Corner`, `Block`, and `Ignore`.
option	Specifies the name of the option to set. You can select from the following list: Pin Options Extract Options Abstract Options Verify Options
value	Specifies the value for the specified option. For the list of valid values for each of the above listed options, refer to the respective option details.

Value Returned

`t`	Returns `t` if the function executes successfully.
`nil`	Returns `nil` if an error is encountered.

Example

```
absSetBinOption "Core" "AbstractAdjustBoundaryPinsPwr" "true"
=> t
```

Sets the abstract option `AbstractAdjustBoundaryPinsPwr` on.

© 2020 All Rights Reserved.

Flow Functions

Use the following functions to execute flow steps in Abstract Generator.

- absPins
- absExtract
- absAbstract
- absVerify

© 2020 All Rights Reserved.

absPins

```
absPins(
     )
     => 0 / 1
```

Description

Runs the Pins step for the selected cells and creates a pins view for those cells based on options set using `absSetBinOption`.

Arguments

None

Value Returned

`0`	Returns `0` if an error is encountered.
`1`	Returns `1` if the function executes successfully.

Associated Options

[AbstractRemoveRedundantBlockages { true | false }]
[PinsTextPinMap *text_pin_spec*]
[PinsPowerNames *power_pin_names*]
[PinsGroundNames *ground_pin_names*]
[PinsClockNames *clock_pin_names*]
[PinsAnalogNames *analog_pin_names*]
[PinsOutputNames *output_pin_names*]
[PinsTextPromoteLevel *integer*]
[PinsGeomSearchLevel *integer*]
[PinsTextManipulation *text_to_remove text_to_replace*]
[PinsTextPreserveLabels { true | false }]
[PinsBoundaryCreate { off | always | as needed }]
[PinsBoundaryLayers *layers*]
[PinsCreatePolyPRB { true | false }]
[PinsBoundarySizeLeft *value*]
[PinsBoundarySizeRight *value*]
[PinsBoundarySizeTop *value*]
[PinsBoundarySizeBottom *value*]

© 2020 All Rights Reserved.

[PinsBoundaryFixedLeft *value*]
[PinsBoundaryFixedRight *value*]
[PinsBoundaryFixedTop *value*]
[PinsBoundaryFixedBottom *value*]
[PinsPreserveRoutingBlockages { true | false }]
[PinsCreatePwrPinsFromRouting { true | false }]
[PinsPwrRoutingLayers *layers*]
[PinsRestrictToPRBndry { true | false }]
[PinsFromTextForExistingPins { true | false }]

© 2020 All Rights Reserved.

Option Descriptions

`AbstractRemoveRedundantBlockages { true | false }`

Controls whether Abstract Generator removes redundant blockages, where the shapes of the same layer coincide with each other.

- If `true`, Abstract Generator removes redundant blockages. If `PinsPreserveRoutingBlockages` is also `true`, local blockages in the layout of the cell are preserved.
- If `false`, Abstract Generator does not remove redundant blockages.

`PinsTextPinMap` *`text_pin_spec`*

Maps text labels in the layout view to pins on nets in the pins view.

`PinsPowerNames` *`power_pin_names`*

Identifies power pin names based on label strings in the abstract. Enter a list of possible names for the power pins. This can be a list of regular expressions, each separated by a space.

`PinsGroundNames` *`ground_pin_names`*

Identifies ground pin names based on label strings in the abstract. Enter a list of possible names for the ground pins. This can be a list of regular expressions, each separated by a space.

`PinsClockNames` *`clock_pin_names`*

Identifies clock pin names based on label strings in the abstract. Enter a list of possible names for the clock pins. This can be a list of regular expressions, each separated by a space.

`PinsAnalogNames` *`analog_pin_names`*

Identifies analog pin names based on label strings in the abstract. Enter a list of possible names for the analog pins. This can be a list of regular expressions, each separated by a space.

`PinsOutputNames` *`output_pin_names`*

© 2020 All Rights Reserved.

Identifies output pin names based on label strings in the abstract. Enter a list of possible names for the output pins. This can be a list of regular expressions, each separated by a space.

`PinsTextPromoteLevel` *`integer`*

Controls how far down the design hierarchy Abstract Generator searches for text labels during text-to-pin mapping.

`PinsGeomSearchLevel` *`integer`*

Controls how far down the design hierarchy Abstract Generator searches for metal underneath a text label.

`PinsTextManipulation` *`text_to_remove text_to_replace`*

Uses regular expressions to locate unnecessary text in labels and either remove it completely or replace it with the text you require.

Every odd regular expression represents *`text_to_remove`*, while every even regular expression represents *`text_to_replace`*.

To remove text completely, use `{}`as the *`text_to_replace`* element.

`PinsTextPreserveLabels { true | false }`

Preserves text labels from the layout view in the final abstract.

`PinsBoundaryCreate { off | always | as needed }`

© 2020 All Rights Reserved.

Specifies whether Abstract Generator is to create a place-and-route boundary based on the layers supplied for the `PinsBoundaryLayers` argument.

- `off` means that Abstract Generator never creates a place-and-route boundary and reports an error if the boundary is missing from the layout.
- `always` means that Abstract Generator always creates a place-and-route boundary, after deleting any already existing boundary. It checks for the existence of a boundary on LPP `prboundary boundary`, deletes if one is found, and then creates the new boundary. If none exists on LPP `prboundary boundary`, it then searches for an existing boundary on LPP `prboundary drawing`, and deletes if any is found before creating the new boundary.
- `as needed` means that Abstract Generator creates a place-and-route boundary only if one does not already exist on LPP `prboundary boundary`.

The new place-and-route boundary created is rectangular. To allow for the creation of a rectilinear place-and-route boundary, use the `PinsCreatePolyPRB` option.

`PinsBoundaryLayers` *`layers`*

Specifies the layers to be used in calculating the place-and-route boundary. The boundary is drawn so that it encloses all the geometry found on these layers.

The argument is valid only if you set `PinsBoundaryCreate` to `always` or `as needed`.

`PinsCreatePolyPRB { true | false }`

Specifies the creation of a rectilinear place-and-route boundary. By default, it is set to `false`, in which case a rectangular place-and-route boundary is created.

The argument is valid only if you set `PinsBoundaryCreate` to `always` or `as needed`.

`PinsBoundarySizeLeft` *`value`*

© 2020 All Rights Reserved.

Specifies the distance in microns to be added to the left boundary edge. This is one of four boundary options to let you increase or decrease the size of an existing or calculated place and route boundary.

Entering a positive number for this option moves the boundary further to the left by that amount; i.e., the lower left coordinate of the boundary is decreased.

This option is exclusive with the `PinsBoundaryFixedLeft` option.

`PinsBoundarySizeRight` *`value`*

Specifies the distance in microns to be added to the right boundary edge. This is one of four boundary options to let you increase or decrease the size of an existing or calculated place and route boundary.

This option is exclusive with the `PinsBoundaryFixedRight` option.

`PinsBoundarySizeTop` *`value`*

Specifies the distance in microns to be added to the top boundary edge. This is one of four boundary options to let you increase or decrease the size of an existing or calculated place and route boundary.

This option is exclusive with the `PinsBoundaryFixedTop` option.

`PinsBoundarySizeBottom` *`value`*

Specifies the distance in microns to be added to the bottom boundary edge. This is one of four boundary options to let you increase or decrease the size of an existing or calculated place and route boundary.

This option is exclusive with the `PinsBoundaryFixedBottom` option.

`PinsBoundaryFixedLeft` *`value`*

Fixes the left boundary edge of a cell to the specified value.

This option is exclusive with the `PinsBoundarySizeLeft` option.

© 2020 All Rights Reserved.

`PinsBoundaryFixedRight` *`value`*

Fixes the right boundary edge of a cell to the specified value.

This option is exclusive with the `PinsBoundarySizeRight` option.

`PinsBoundaryFixedTop` *`value`*

Fixes the top boundary edge of a cell to the specified value.

This option is exclusive with the `PinsBoundarySizeTop` option.

`PinsBoundaryFixedBottom` *`value`*

Fixes the bottom boundary edge of a cell to the specified value.

This option is exclusive with the `PinsBoundarySizeBottom` option.

`PinsPreserveRoutingBlockages { true | false }`

Controls how Abstract Generator processes local blockages defined in the `BLOCKAGE` section in DEF 5.4. Local blockages do not have the `+PUSHDOWN`, `+FILL`, or `+SLOT` modifiers.

- If `false`, Abstract Generator removes any local blockages and proceeds as if they had never been specified in the DEF file.
- If `true`, and if `AbstractRemoveRedundantBlockages` is also `true`, Abstract Generator preserves local blockages along with their attributes from level 0 to level 32. The preserved layout blockages exists in the abstract view along with the other blockages generated in the abstract step.

Note: To preserve the routing blockages of layout, you need to delete the `boundary` purpose from the `ExcludePurposeList` option. For this, set the absSetOption as `absSetOption (ExcludePurposeList " ")`.

© 2020 All Rights Reserved.

`PinsCreatePwrPinsFromRouting { true | false }`

Controls whether pins are created from the routing defined in the `SPECIALNETS` section of the DEF file.

- If `false`, Abstract Generator creates pins only from the pin geometry defined in the `PINS` section of the DEF file. If there is no pin geometry, the routing is turned into pins instead.
- If `true`, Abstract Generator creates pins from the routing defined in the `SPECIALNETS` section for the layers specified for the `PinsPwrRoutingLayers` option. It does this even if there are pins already defined in the `PINS` section.

When creating pins, Abstract Generator ignores `FILLWIRE` and `SHIELD` net segments. It also creates pins in instances that are logically connected to the top-level nets.

`PinsPwrRoutingLayers` *`layers`*

Specifies the layers to be used in the creation of pins from routing defined in the `SPECIALNETS` section of the DEF file.

`PinsRestrictToPRBndry { true | false }`

Controls the geometry of N-Well pins created during the abstract generation process. If set to t, the size of N-Well pins is restricted to the PRBoundary.

`PinsFromTextForExistingPins { true | false }`

When set to `true`, Abstract Generator generates additional pins from labels, even if the corresponding net has existing pins in the layout. The default value is false.

```
absSetBinOption( "Block"
"PinsFromTextForExistingPins" "true")
```

Example

```
absSetBinOption "Core" "PinsTextPinMap" "(Metal1 (Metal1 drawing)) (Metal2 (Metal2
drawing))"
=> t
```


© 2020 All Rights Reserved.

```
absPins
=> 1
```

Specifies a text-to-pin map, for the `Core` bin, where labels on layer `Metal1` (all purposes) are to be mapped first to shapes on layer `Metal1` (purpose `drawing` only). If no shapes are found on `Metal1 drawing`, then the labels are to be mapped to shapes on layer `Metal2` (purpose `drawing` only). Finally, runs the Pins step.

© 2020 All Rights Reserved.

absExtract

```
absExtract(
     )
     => 0 / 1
```

Description

Runs the Extract step for the selected cells and creates an extracted view for those cells based on the options set using `absSetBinOption`.

Arguments

None

Value Returned

`0`	Returns `0` if an error is encountered.
`1`	Returns `1` if the function executes successfully.

Associated Options

[ExtractSig { true | false }]
[ExtractAntennaPinPathsOnly { true | false }]
[AbstractExtractGSpecTable *geom_spec*]
[ExtractLayersSig *layers*]
[ExtractLayersSigWeak { Strong | Weak }]
[ExtractPinLayersSig *layers*]
[ExtractNumLevelsSig *max_depth*]
[ExtractDistSig *distance*]
[ExtractWidthSig *width*]
[ExtractMustConnectAllPinsAlways { true | false }]
[ExtractMustConnectTerminals *terminals*]
[ExtractMustJoinAlways { true | false }]
[ExtractMustJoinTerminals *terminals*]
[ExtractPwr { true | false }]
[AbstractExtractPwrGSpecTable *geom_spec*]
[ExtractLayersPwr *layers*]
[ExtractPinLayersPwr *layers*]
[ExtractNumLevelsPwr *max_depth*]

© 2020 All Rights Reserved.

[ExtractDistPwr *distance*]
[ExtractWidthPwr *width*]
[ExtractAntennaHier { true | false }]
[ExtractAntennaSizeInput { true | false }]
[ExtractAntennaSizeOutput { true | false }]
[ExtractAntennaSizeInout { true | false }]
[ExtractAntennaMetalArea { true | false }]
[ExtractAntennaMetalSideArea { true | false }]
[ExtractAntennaMaxCutCARLowestCut { true | false }]
[AbstractAntennaGSpecTable *geom_spec*]
[ExtractAntennaGate *geom_spec*]
[ExtractAntennaDrain *geom_spec*]
[ExtractAntennaExcludeCumPerLayerAntennaRatio { true | false }]
[ExtractAntennaOxide { Oxide1 | … | Oxide32 }]
[ExtractDiffAntennaLayers { true | false }]
[ExtractAntennaLayers *layers*]
[ExtractAntennaGSpecTable *geom_spec*]
[ExtractConnectivity *layers*]
[ExtractAntennaIncludePolyCAR { true | false }]

© 2020 All Rights Reserved.

Option Descriptions

`ExtractSig { true | false }`

Specifies that connectivity information be extracted for signal nets, when set to `true`.

If `ExtractSig` is set to `false`, all other signal arguments are invalid except the `MustJoin` arguments.

`ExtractAntennaPinPathsOnly { true | false }`

Specifies that only pin nodes and pin-connected nodes be considered for antenna number generation, when set to `true`. In this case, `ExtractSig` should be set to `false`.

If set to `false` or if `ExtractSig` is set to `true`, all nodes are considered for antenna number generation.

`AbstractExtractGSpecTable` *`geom_spec`*

Specifies a geometric operation to establish the geometry or shapes that should be present on a particular layer when extraction is performed.

For information about creating a geometry specification, see the Layer Assignment for Signal Extraction table.

`ExtractLayersSig` *`layers`*

Specifies the layers through which the extractor extracts each net. To specify this argument, first set the `ExtractSig` option to `true`.

`ExtractLayersSigWeak { Strong | Weak }`

Selects the connectivity setting for a particular layer.

This argument is valid only if you have provided the associated layer and geometry specification.

`ExtractPinLayersSig` *`layers`*

© 2020 All Rights Reserved.

Determines which of the shapes found by the extractor should be turned into pins.

This argument is valid only if the associated layer, geometry specification and connectivity have been provided.

`ExtractNumLevelsSig` *`max_depth`*

Controls how far down the design hierarchy the extractor searches for shapes. The default is `32`, the maximum depth of hierarchy in the database.

`ExtractDistSig` *`distance`*

Restricts the shapes considered by the extractor to those that intersect a bounding box around each starting pin shape. Specify the size of the bounding box in microns.

`ExtractWidthSig` *`width`*

Specifies the minimum width of shape to extract. The extractor extracts only shapes that are greater than or equal to the width you supply.

`ExtractMustConnectAllPinsAlways { true | false }`

Establishes a must-connect relation between all disjoint groups of shapes extracted for any net in the layout.

`ExtractMustConnectTerminals` *`terminals`*

Specifies a list of terminal names, each separated by a single space or a comma.

Establishes a must-connect relation between disjoint groups of shapes extracted for the terminals listed. For all pins in the net, Abstract Generator retains the same terminal name.

To specify this argument, set `ExtractMustConnectAllPinsAlways` to `false`.

`ExtractMustJoinAlways { true | false }`

Establishes a must-join relationship between all disjoint groups of geometries extracted from separate starting pins on the same net.

© 2020 All Rights Reserved.

`ExtractMustJoinTerminals` *`terminals`*

Specifies a list of terminal names, each separated by a single space or a comma.

If the extractor finds a terminal name matching one of the names supplied, and if there are disjoint geometries from different starting pins on that net after extraction, then Abstract Generator creates a `MUSTJOIN` relationship for that terminal. To specify this argument, set the `ExtractMustJoinAlways` to `false`.

`ExtractPwr { true | false }`

Specifies whether or not to extract connectivity for power nets.

`AbstractExtractPwrGSpecTable` *`geom_spec`*

Specifies a geometric operation to establish the geometry and shapes that should be present on a particular layer when extraction is performed.

For information about creating a geometry specification, see the Layer Assignment for Power Extraction table.

`ExtractLayersPwr` *`layers`*

Specifies the layers through which the extractor extracts each net. To specify this argument, first set the `ExtractPwr` option to `true`.

`ExtractPinLayersPwr` *`layers`*

Determines which of the shapes found by the extractor should be turned into pins.

This argument is valid only if the associated layer and geometry specification have been provided.

`ExtractNumLevelsPwr` *`max_depth`*

Controls how far down the design hierarchy the extractor searches for shapes. The default is `32`, the maximum depth of hierarchy in the database.

`ExtractDistPwr` *`distance`*

© 2020 All Rights Reserved.

Restricts the shapes considered by the extractor to those that intersect a bounding box around each starting pin shape. Specify the size of the bounding box in microns.

`ExtractWidthPwr` *`width`*

Specifies the minimum width of shape to extract. The extractor extracts only shapes that are greater than or equal to the width you supply.

`ExtractAntennaHier { true | false }`

Calculates antenna hierarchically by summing the antenna data on standard abstracts.

`ExtractAntennaSizeInput { true | false }`

Calculates either the `ANTENNASIZE` LEF (if you are using LEF 5.3) or the `ANTENNAGATEAREA` attribute (if you are using LEF 5.4) for input pins.

`ExtractAntennaSizeOutput { true | false }`

Calculates either the `ANTENNASIZE` LEF (if you are using LEF 5.3) or the `ANTENNAGATEAREA and ANTENNADIFFAREA` attributes (if you are using LEF 5.4) for output pins.

`ExtractAntennaSizeInout { true | false }`

Calculates either the `ANTENNASIZE` LEF (if you are using LEF 5.3) or the `ANTENNAGATEAREA and ANTENNADIFFAREA` attributes (if you are using LEF 5.4) for IO pins.

`ExtractAntennaMetalArea { true | false }`

Calculates the area of metal that will be connected to the pin when the layer is fabricated.

`ExtractAntennaMetalSideArea { true | false }`

Calculates the side area of the metal connected to the pins is calculated (perimeter x thickness).

`ExtractAntennaMaxCutCARLowestCut { true | false }`

© 2020 All Rights Reserved.

Includes `CUT01` layer for the calculation of `ANTENNAMAXCUTCAR`. For more information about `ANTENNAMAXCUTCAR` calculation, see Performing Process Antenna Calculations.

`AbstractAntennaGSpecTable` *`geom_spec`*

Specifies a geometric operation to define the gate or drain geometry regions. For information about creating a geometry specification, see the Layer Assignment for Antenna Regions table.

`ExtractAntennaGate` *`geom_spec`*

Specifies a geometric operation to find the gate geometry; for example,

```
(Poly (Poly and Oxide))
```

`ExtractAntennaDrain` *`geom_spec`*

Specifies a geometric operation to determine the drain area that must be removed from the area of the diffusion; for example,

```
(Oxide (Oxide andnot Poly))
```

`ExtractAntennaExcludeCumPerLayerAntennaRatio { true | false }`

Extracts antenna factors by using the `antenna` constraint when set to `true`, its default value, or not set. When set to `false`, antenna factors are extracted using `cumPerLayerAntennaRatio`.

`ExtractAntennaOxide { Oxide1 | … | Oxide32 }`

Selects an oxide model from the available oxide types, which are in the range `Oxide1` through `Oxide32`. For example:

```
("Core" "ExtractAntennaOxide" "(OD Oxide12)")
```

`ExtractDiffAntennaLayers { true | false }`

Sets different layer assignments for antenna extraction.

`ExtractAntennaLayers` *`layers`*

© 2020 All Rights Reserved.

Specifies the layers for which you want to extract antenna data.

This option is valid only if you are using LEF 5.4 and the `ExtractDiffAntennaLayers` option is set to `true`.

For information about creating a geometry specification, see the *Layer Assignment for Antenna Extraction* table.

`ExtractAntennaGSpecTable` *`geom_spec`*

Specifies a geometric operation to define antenna regions.

For information about creating a geometry specification, see the Layer Assignment for Antenna Regions table.

`ExtractConnectivity` *`layers`*

Specifies layers for which connectivity is to be extracted.

`ExtractAntennaIncludePolyCAR { true | false }`

Includes poly shape area in the calculation of `ANTENNAPARTIALMETALAREA`.

Example

```
absSetBinOption "Core" "ExtractPwr" "true"
=> t
absSetBinOption "Core" "ExtractLayersPwr" "Metal1 Metal2 Metal3"
=> t
absExtract
=> 1
```

Specifies that power nets should be extracted for `Metal1`, `Metal2`, and `Metal3` layers for cells in the core bin and then runs the Extract step.

© 2020 All Rights Reserved.

absAbstract

```
absAbstract(
     )
     => 0 / 1
```

Description

Runs the Abstract step for selected cells and creates an abstract view for those cells based on the options set using `absSetBinOption`.

Arguments

None

Value Returned

`0`	Returns `0` if an error is encountered.
`1`	Returns `1` if the function executes successfully.

Associated Options

[AbstractAdjustBoundaryPinsSig { true | false }]
[AbstractAdjustBoundaryPinsPwr { true | false }]
[AbstractAdjustBoundaryPinsSigDist *distance*]
[AbstractAdjustBoundaryPinsPwrDist *distance*]
[AbstractAdjustRingPinsPwr { true | false }]
[AbstractAdjustRingPinsDist *distance*]
[AbstractAdjustFollowRingPin { true | false }]
[AbstractAdjustPowerGeometryGroups { single | separate | overlap }]
[AbstractAdjustPowerGeometryGroupsEastWest { single | separate | overlap | none }]
[AbstractAdjustEdgeTowardsCore]
[AbstractAdjustClassCoreNets *net_name*]
[AbstractAdjustIncludeAllShapes { true | false }]
[AbstractAdjustCopyClassCorePort { true | false }]
[AbstractAdjustPinsTouchBoundary { true | false }]
[AbstractAdjustPreserveViasPwr { true | false }]
[AbstractAdjustPowerRailOp *net_name*]
[AbstractAdjustPowerRailOpTable *net_spec*]
[AbstractAdjustPowerRailWidth *width*]

© 2020 All Rights Reserved.

[AbstractAdjustAllowPin *geom_spec*]
[AbstractAdjustAvoidPin *geom_spec*]
[AbstractAdjustAllowPowerPin *geom_spec*]
[AbstractAdjustAvoidPowerPin *geom_spec*]
[AbstractBlockageTable *layer_names*]
[AbstractBlockageDetailedLayers *layer_names*]
[AbstractBlockageCoverLayers *layer_names*]
[AbstractBlockageShrinkWrapLayers *layer_names*]
[AbstractBlockageCutAroundPin]
[AbstractBlockageMaxRoutingSpace]
[AbstractBlockageCorridorCut]
[AbstractBlockagePinCutWindow *distance*]
[AbstractBlockageDownPinCutWindow *distance*]
[AbstractBlockageCutForAboveLayers]
[AbstractBlockageShrinkAdjust *distance*]
[AbstractBlockageShrinkTracks *number_of_tracks*]
[AbstractBlockageUserDefinedDistanceForAddBack *distance*]
[AbstractBlockageUserDefinedWidth *width*]
[AbstractBlockageCoverLayersDist *distance*]
[AbstractZeroSpacingBlockage { true | false }]
[AbstractDensity { true | false }]
[AbstractDensityUseSignalLayers { true | false }]
[AbstractDensityUseAntennaLayers { true | false }]
[AbstractDensityUsePwrLayers { true | false }]
[AbstractDensityLayers *layer_name geom_spec*]
[AbstractDensityWindowWidth *layer_name value*]
[AbstractDensityWindowHeight *layer_name value*]
[AbstractDensityDefaultWindowWidth *value*]
[AbstractDensityDefaultWindowHeight *value*]
[AbstractPinFracture { true | false }]
[AbstractBlockageFracture { true | false }]
[AbstractAdjustStairStepCover { full | truncate | partial }]
[AbstractAdjustStairStepWidth *width*]
[AbstractSiteName *site_name*]
[AbstractSiteNameDefine *site_name*]
[AbstractSiteArrayPattern { true | false }]
[AbstractSiteArrayCells *sub-cellview site*]
[AbstractOverlapLayerAction { off | always | as needed }]
[AbstractOverlapLayers *layers*]
[AbstractOverlapLayerSize *size*]
[AbstractOverlapLayerSmoothFactor *size*]

© 2020 All Rights Reserved.

[AbstractGridMode { report | calculate | calculate M1 | calculate M2 | off }]
[AbstractUpdateTechFile { true | false }]
[AbstractMetal1Pitch *value*]
[AbstractMetal1Offset *value*]
[AbstractMetal2Pitch *value*]
[AbstractMetal2Offset *value*]
[AbstractMetal1PitchPercent *value*]
[AbstractMetal2PitchPercent *value*]
[AbstractDiagonalVias { true | false }]
[AbstractGridDistanceMetric { maxxy | euclidean }]
[AbstractPinStretchOutPrb { true | false }]
[AbstractBlockageEXCEPTPGNET]
[AbstractTopLayerCoverBlockage {Routing Layer}]
[AbstractBlockageUserDefinedSpacing]
[AbstractFractureRectilinearShapes { true | false }]
[BlockageCutVia]
[BlockageLargePurposeList]
[BlockageLargeShape]
[BlockageLargeShapePct]

© 2020 All Rights Reserved.

Option Descriptions

`AbstractAdjustBoundaryPinsSig { true | false }`

Makes the signal pin shapes created during extraction square.

`AbstractAdjustBoundaryPinsPwr { true | false }`

Makes the power pin shapes created during extraction square.

`AbstractAdjustBoundaryPinsSigDist` *`distance`*

Specifies the distance (in microns) at which the signal net geometries should be from the boundary so that they can be considered as boundary pins. The geometries that are within the specified distance from the boundary are squared.

By default, the shape(s) closest to the boundary is squared.

`AbstractAdjustBoundaryPinsPwrDist` *`distance`*

Specifies the distance (in microns) at which the power net geometries should be from the boundary so that they can be considered as boundary pins. The geometries that are within the specified distance from the boundary are squared.

By default, the shape(s) closest to the boundary is squared.

`AbstractAdjustRingPinsPwr { true | false }`

Creates ring pins where any shapes in a power net are found to form a ring.

This argument is valid only when processing blocks and is mutually exclusive with the `AbstractAdjustBoundaryPinsPwr` argument.

`AbstractAdjustRingPinsDist` *`distance`*

© 2020 All Rights Reserved.

Specifies in microns the maximum distance from the boundary that extracted power net geometry can be in order for it to be considered part of the ring and therefore to be included in the abstract as pin geometry.

This argument is valid only if you are processing a block and if `AbstractAdjustRingPinsPwr` is `true`.

`AbstractAdjustFollowRingPin { true | false }`

Specifies that all pin shapes, including the ones that do not form the ring, are considered as pins in the final abstract.

If the option is set to `true` and no value is specified for the `AbstractAdjustRingPinsDist` option, only the pin shapes whose distance from the boundary is the same or less than the distance of the farthest edge of the ring pin shape are included in the abstract. If the option is set to `true` and a value is specified for the `AbstractAdjustRingPinsDist` option, pin shapes are included in the abstract only if their distance from the boundary is within the distance specified for the option.

Specify an arbitrarily large value for the `AbstractAdjustRingPinsDist` option so that no filtering happens based on the distance you specify.

`AbstractAdjustPowerGeometryGroups { single | separate | overlap }`

Controls how power shapes are grouped into LEF PORTS.

- `single` creates a single LEF PORT for all the power shapes.
- `separate` creates a separate LEF PORT for each power shape.
- `overlap` forms LEF PORTS from groups of overlapping power pins. This means that the power shapes that overlap go into the same LEF PORT.

`AbstractAdjustPowerGeometryGroupsEastWest { single | separate | overlap | none }`

© 2020 All Rights Reserved.

Controls how power shapes touching the east or west edge of a cellview are grouped into LEF PORTS. The remaining shapes that do not touch the east or west side are grouped according to the *AbstractAdjustPowerGeometryGroups* option.

- `single` creates a single LEF PORT for all the power shapes touching the east or west edge.
- `separate` creates a separate LEF PORT for each power shape touching the east or west edge.
- `overlap` forms LEF PORTS from groups of overlapping power pins touching the east or west edge. This means that the power shapes that overlap go into the same LEF PORT.

By default, this option is set to `None`

A power shape touching the east or west edge as well as one of either the north or the south edge is grouped as per the *AbstractAdjustPowerGeometryGroupsEastWest* option. Even a shape touching the north and south edges in addition to touching either the east or the west edge is grouped as per the `AbstractAdjustPowerGeometryGroupsEastWest` option.

`AbstractAdjustEdgeTowardsCore`

Specifies the direction of the CORE-facing edge. This argument is available only when a pad cell in the IO bin is being processed. You can specify multiple directions of the CORE-facing edge. The valid values for the `CORE`-facing edge are, north, south, east, west, or a combination of any of the four core edges.

`AbstractAdjustClassCoreNets` *`net_name`*

Specifies which nets should have `CLASS CORE` ports. Accepts a regular expression; for example,

```
^((V(DD|CC))|(v(dd|cc)))(!)?$
```

`AbstractAdjustIncludeAllShapes { true | false }`

© 2020 All Rights Reserved.

Specifies that multiple ports that are equidistant from the core-facing edge of a pad are considered as `CLASS CORE` ports. Only the geometries with their center lines perpendicular to the core-facing edge are considered. This option is applicable to pad cells that are processed in the IO bin.

This option is set to `false` by default, which means that only the geometry closest to the core-facing edge becomes a `CLASS CORE` port. When you set the option to `true`, all the geometries equidistant from the core-facing edge are considered as `CLASS CORE` ports.

`AbstractAdjustCopyClassCorePort { true | false }`

Duplicates any geometry found to belong to a `CLASS CORE` port on the net (set using the `AbstractAdjustClassCoreNets` argument) in a non-`CLASS CORE` port in the exported abstract LEF.

`AbstractAdjustPinsTouchBoundary { true | false }`

Specifies that only pins that touch the boundary of the cell become CLASS CORE ports in LEF.

`AbstractAdjustPreserveViasPwr { true | false }`

Preserves existing layouts rather than removing them when the `absAbstract` function is run.

`AbstractAdjustPowerRailOp` *`net_name`*

Specifies the net name whose power and ground rail characteristics you want to set.

`AbstractAdjustPowerRailOpTable` *`net_spec`*

© 2020 All Rights Reserved.

Specifies the characteristics of the power and ground rails for the current bin.

The *net_spec* takes the form

netName shape width offset

- *netName* is the name of a net.
- *shape* can be one of calculate, feedthru, abutment or None
- *width* is the rail width in microns (must be non-negative float)
- *offset* is the rail offset in microns (any float)

AbstractAdjustPowerRailWidth *width*

Sets the rail width value for the net specified by AbstractAdjustPowerRailOp.

AbstractAdjustAllowPin *geom_spec*

Specifies a geometric operation to describe the regions where pins are to be allowed. If the AllowAvoidSignalAndPowerPinsSeparately Shell environment variable is set, this option is used only for signal pins. If not set, it is used for both signal and power pins.

AbstractAdjustAvoidPin *geom_spec*

Specifies a geometric operation to describe the regions where pins are to be avoided. If the AllowAvoidSignalAndPowerPinsSeparately Shell environment variable is set, this option is used only for signal pins. If not set, it is used for both signal and power pins.

AbstractAdjustAllowPowerPin *geom_spec*

Specifies a geometric operation to describe the regions where only power pins are to be allowed. This option works only if the AllowAvoidSignalAndPowerPinsSeparately Shell environment variable is set.

AbstractAdjustAvoidPowerPin *geom_spec*

© 2020 All Rights Reserved.

Specifies a geometric operation to describe the regions where only power pins are to be avoided. This option works only if the `AllowAvoidSignalAndPowerPinsSeparately` Shell environment variable is set.

`AbstractBlockageTable` *`layer_names`*

Specifies the layers on which to create blockages. This controls the final blockage geometry in the abstract.

`AbstractBlockageDetailedLayers` *`layer_names`*

Specifies the layers for which a detailed blockage model is to be created. The detailed blockage model generates blockages only where there are real shapes in the cell on the layer.

`AbstractBlockageCoverLayers` *`layer_names`*

Specifies the layers for which a cover blockage model is to be created. A cover blockage blocks the entire area that is not occupied by pin shapes, effectively blocking a layer for routing

`AbstractBlockageShrinkWrapLayers` *`layer_names`*

Specifies the layers for which a shrink wrap blockage model is to be generated. The shrink wrap blockage model fills in smaller, less useful free spaces and leaves larger spaces in the cell open. This allows over-the-cell routing without modeling each obstruction individually.

`AbstractBlockageCutAroundPin`

Controls whether the blockage around a pin is cut for allowing the router to access the pin. If this option is enabled, the blockage is cut at `min-spacing` distance around pins.

`AbstractBlockageMaxRoutingSpace`

© 2020 All Rights Reserved.

Controls whether blockage is to be created around a block to drive the external routes maximum distance away from the block.

This option is valid only if `AbstractBlockageCutAroundPin` has been enabled. When both `AbstractBlockageCutAroundPin` and `AbstractBlockageMaxSpacingRoute` are enabled, the `AbstractBlockageCorridorCut` option is ineffective even if enabled.

`AbstractBlockageCorridorCut`

Controls whether corridors are cut out around pins for allowing routers to access internal pins.

This option is valid only if `AbstractBlockageCutAroundPin` has been enabled. This option is ineffective when both `AbstractBlockageCutAroundPin` and `AbstractBlockageMaxSpacingRoute` have been enabled.

`AbstractBlockagePinCutWindow` *`distance`*

Specifies the distance to cut around pins on the same layer. If set as `0`, the blockage completely covers the design, including the pin shape, and there is no pin cut out.

`AbstractBlockageDownPinCutWindow` *`distance`*

Specifies the distance to cut around pins on the layer below.

`AbstractBlockageCutForAboveLayers`

Specifies a valid metal layer that contains pins and the size of the pin cutouts to be created in the blockage layers above this layer. For example,

```
absSetBinOption("Block"
"AbstractBlockageCutForAboveLayers" "Metal2
(Metal4 0.2)")
```

`AbstractBlockageRoutingChannelOnLayers`

© 2020 All Rights Reserved.

Specifies the shape of the routing channel to be created when generating the cover blockage. For example,

```
absSetBinOption ("Block"
"AbstractBlockageRoutingChannelOnLayers"
"(Metal1 (y0 drawing)) (Metal2 (y0 drawing))")
```

`AbstractBlockageShrinkAdjust` *`distance`*

Specifies a minimum distance in microns between blockages. Blockages separated by a distance less than or equal to the specified distance are merged.

This option is valid only for layers specified by `AbstractBlockageShrinkWrapLayers`.

`AbstractBlockageShrinkTracks` *`number_of_tracks`*

Specifies the minimum number of tracks between blockages. Blockages separated by a distance less than or equal to the specified number of tracks are merged.

This option is valid only for layers specified by `AbstractBlockageShrinkWrapLayers`.

`AbstractBlockageUserDefinedDistanceForAddBack` *`distance`*

Specifies the distance within which the geometries need to be added back when Abstract Generator creates a cover blockage with pin cutout.

`AbstractBlockageUserDefinedWidth` *`width`*

Sets the effective width attribute for layer blockages. This value can be set only if the spacing value is set to zero. The `AbstractBlockageUserDefinedWidth` option takes only positive float values, for example:

```
absSetBinOption( "Block"
"AbstractBlockageUserDefinedWidth"
"(Metal1 5.0)  (Metal2 4.0)")
```

`AbstractBlockageCoverLayersDist` *`distance`*

© 2020 All Rights Reserved.

Specifies the minimum distance between a cover blockage and the cell boundary. If you do not specify a distance, Abstract Generator uses a distance equal to one half of the minimum layer separation.

This option is valid only for layers specified by `AbstractBlockageCoverLayers`.

`AbstractZeroSpacingBlockage { true | false }`

By default, the option is set to `false`. The working of this option depends on the value set for the option `AbstractBlockagePinCutWindow`:

- `AbstractBlockagePinCutWindow` is non-zero or is not specified

  In this case, the following situations are possible:

  - `AbstractZeroSpacingBlockage` is `true`: The blockage around the pin is cut out at zero spacing from the pin.
  - `AbstractZeroSpacingBlockage` is `false`: The blockage around the pin is cut out at default spacing from the pin.

- `AbstractBlockagePinCutWindow` is `0`

  If set as `0`, this option overrides any value set for the option `AbstractZeroSpacingBlockage`. In this scenario, the blockage completely covers the design, including the pin shape, and there is no pin cut out.

`AbstractDensity { true | false }`

Specifies whether Abstract Generator is to generate the metal density information. The default value of this option is `false`.

`AbstractDensityUseSignalLayers { true | false }`

© 2020 All Rights Reserved.

Directs Abstract Generator to use the same geometry specifications for metal density calculation as are specified for signal extraction using the `ExtractLayersSig` option.

The default value of `AbstractDensityUseSignalLayers` is `false`.

Setting this option to `true` automatically sets the `AbstractDensityUsePwrLayers` and `AbstractDensityUseAntennaLayers` options to `false`.

`AbstractDensityUseAntennaLayers { true | false }`

Directs Abstract Generator to use the same geometry specifications for metal density calculation as will be used for process antenna calculation. In other words, to specify the layers and their geometry specifications, use the `ExtractAntennaLayers` option if the `ExtractDiffAntennaLayers` option is set to `true`; otherwise use the `ExtractLayersSig` option.

The default value of `AbstractDensityUseAntennaLayers` is `false`.

Setting this option to `true` automatically sets the `AbstractDensityUseSignalLayers` and `AbstractDensityUsePwrLayers` options to `false`.

`AbstractDensityUsePwrLayers { true | false }`

Directs Abstract Generator to use the same geometry specifications for metal density calculation as are specified for power net extraction using the `ExtractLayersPwr` option.

The default value of `AbstractDensityUsePwrLayers` is `false`.

Setting this option to `true` automatically sets the `AbstractDensityUseSignalLayers` and `AbstractDensityUseAntennaLayers` options to `false`.

`AbstractDensityLayers` *`layer_name geom_spec`*

© 2020 All Rights Reserved.

Specifies the exclusive metal layers along with their geometry specifications that are to be considered by Abstract Generator for metal density calculation.

For example,

```
AbstractDensityLayers "Metal1 Metal2
Metal3 (Metal4 (Metal4 drawing)) Metal5
Metal6"
```

The default value for this option is a list of all metal layers:

```
AbstractDensityLayers "Metal1 Metal2
Metal3 Metal4 Metal5 Metal6"
```

`AbstractDensityWindowWidth` *`layer_name value`*

Specifies the width of the density window for a particular layer. For example,

```
AbstractDensityWindowWidth "(Metal1
10.0)"
```

If you do not specify a value for a layer, the value specified for the `AbstractDensityDefaultWindowWidth` option is taken as the default.

`AbstractDensityWindowHeight` *`layer_name value`*

Specifies the height of the density window for a particular layer. For example,

```
AbstractDensityWindowHeight "(Metal1
10.0)"
```

If you do not specify a value for a layer, the value specified for the `AbstractDensityDefaultWindowHeight` option is taken as the default.

`AbstractDensityDefaultWindowWidth` *`value`*

© 2020 All Rights Reserved.

Specifies the default width of the density window when no width value is specified for a particular layer. For example,

```
AbstractDensityDefaultWindowWidth "30.0"
```

You can override the default value (`20.0`) for the width of the density window by using this option.

`AbstractDensityDefaultWindowHeight` *`value`*

Specifies the default height of the density window when no height value is specified for a particular layer. For example,

```
AbstractDensityDefaultWindowHeight
"40.0"
```

You can override the default value (`20.0`) for the height of the density window by using this option.

`AbstractPinFracture { true | false }`

Controls whether Abstract Generator creates each pin shape as a set of maximum rectangles or as a polygon. This lets you model 45-degree pins if your design requires it.

- If `true`, Abstract Generator creates each pin shape as a set of maximum rectangles. This is recommended if the abstracts are to be routed using Silicon Ensemble or Encounter.
- If `false`, Abstract Generator creates each blockage shape as a polygon. This is the default setting and is recommended if the abstracts are to be routed using the IC Shape-Based Router.

`AbstractBlockageFracture { true | false }`

© 2020 All Rights Reserved.

Controls whether Abstract Generator creates each blockage shape as a set of maximum rectangles or as a polygon.

- If `true`, Abstract Generator creates each blockage shape as a set of maximum rectangles. This is recommended if the abstracts are to be routed using Silicon Ensemble or Encounter.
- If `false`, Abstract Generator creates each blockage shape as a polygon. This is the default and is recommended if the abstracts are to be routed using the IC Shape-Based Router.

`AbstractAdjustStairStepCover { full | truncate | partial }`

Controls the degree to which stair-step blockages cover 45-degree geometry.

`AbstractAdjustStairStepWidth` *`width`*

Specifies the width of the blockage geometry to be created to cover 45-degree shapes.

`AbstractSiteName` *`site_name`*

Specifies a site name to be applied to all cells in the current bin.

`AbstractSiteNameDefine` *`site_name`*

Specify the name of a new site to be used by all cells in the current bin. If the site does not already exist, Abstract Generator defines it in the technology file.

`AbstractSiteArrayPattern { true | false }`

Specifies if Abstract Generator is to calculate site pattern for the gate-array sub-cellviews specified in the option `AbstractSiteArrayCells`.

`AbstractSiteArrayCells` *`sub-cellview site`*

Specifies the mapping of sub-cellviews to sites for calculating site patterns.

`AbstractOverlapLayerAction { off | always | as needed }`

© 2020 All Rights Reserved.

Specifies whether Abstract Generator is to create an overlap boundary.

- `always` means that Abstract Generator always creates an overlap boundary. Any existing overlap layer geometry is overwritten.
- `as needed` means that Abstract Generator creates an overlap boundary if it is not already present or if it is present and was previously calculated by Abstract Generator.
- `off` means that Abstract Generator never creates an overlap boundary.

`AbstractOverlapLayers` *`layers`*

Specifies the layers to be used in calculating the overlap layer boundary. The boundary is drawn so that it encloses all the geometry found on these layers.

The argument is valid only if you set `AbstractOverlapLayerAction` to `always` or `as needed`.

`AbstractOverlapLayerSize` *`size`*

Specifies in microns how much an overlap layer is to be grown.

Use this option to ensure that the placement of cells with abutting overlap layers does not lead to geometry contained in these cells causing violations.

`AbstractOverlapLayerSmoothFactor` *`size`*

Controls the shape of the overlap layer produced.

Use this option to create simpler overlap boundary shapes, typically when processing blocks. The abstract generator takes the computed overlap boundary and removes any cut-outs with a maximum dimension less than the specified distance.

`AbstractGridMode { report | calculate | calculate M1 | calculate M2 | off }`

© 2020 All Rights Reserved.

Specifies the grid analysis mode for standard cell bins.

- In `report` mode, you can specify

  ```
  AbstractMetal1Pitch
  AbstractMetal1Offset
  AbstractMetal2Pitch
  AbstractMetal2Offset
  ```

  But not

  ```
  AbstractMetal1PitchPercent
  AbstractMetal2PitchPercent
  ```

- In `calculate` mode, you can specify

  ```
  AbstractMetal1PitchPercent
  AbstractMetal2PitchPercent
  ```

  But not

  ```
  AbstractMetal1Pitch
  AbstractMetal1Offset
  AbstractMetal2Pitch
  AbstractMetal2Offset
  ```

- In `calculate_M1` mode, you can specify

  ```
  AbstractMetal2Pitch
  AbstractMetal2Offset
  AbstractMetal1PitchPercent
  ```

  But not

  ```
  AbsabstractMetal1Pitch
  AbstractMetal1Offset
  AbstractMetal2PitchPercent
  ```

- In `calculate_M2` mode, you can specify

  ```
  AbstractMetal1Pitch
  AbstractMetal1Offset
  AbstractMetal2PitchPercent
  ```

  But not

  ```
  AbstractMetal2Pitch
  AbstractMetal2Offset
  AbstractMetal1PitchPercent
  ```

```
AbstractUpdateTechFile { true | false }
```


© 2020 All Rights Reserved.

Controls whether the technology file is update with the calculated grid.

`AbstractMetal1Pitch` *`value`*

Specifies the `metal1` layer pitch value (for core cells only).

If you run `AbstractGridMode` in `report` mode, these are the values that are used in the report. If you run grid analysis in `calculate` mode, these values are overwritten with the new best grid values for the layers being calculated.

`AbstractMetal1Offset` *`value`*

Specifies the `metal1` layer offset value (for core cells only).

If you run `AbstractGridMode` in `report` mode, these are the values that are used in the report. If you run grid analysis in `calculate` mode, these values are overwritten with the new best grid values for the layers being calculated.

`AbstractMetal2Pitch` *`value`*

Specifies the `metal2` layer pitch value (for core cells only).

If you run `AbstractGridMode` in `report` mode, these are the values that are used in the report. If you run grid analysis in `calculate` mode, these values are overwritten with the new best grid values for the layers being calculated.

`AbstractMetal2Offset` *`value`*

Specifies the `metal2` layer offset value (for core cells only).

If you run `AbstractGridMode` in `report` mode, these are the values that are used in the report. If you run grid analysis in `calculate` mode, these values are overwritten with the new best grid values for the layers being calculated.

`AbstractMetal1PitchPercent` *`value`*

© 2020 All Rights Reserved.

Sets the upper limit for pitches considered during pitch calculation for the `metal1` layer.

This argument is valid only if `AbstractGridMode` is using set to use one of the calculation values.

`AbstractMetal2PitchPercent` *`value`*

Sets the upper limit for pitches considered during pitch calculation for `metal2` layer.

This argument is valid only if `AbstractGridMode` is using set to use one of the calculation values.

`AbstractDiagonalVias { true | false }`

Controls whether or not the pitches calculated will let vias be placed diagonally adjacent to each other.

When this option is `true`, the calculated pitches are larger than they would otherwise be in order to accommodate diagonal via placement.

`AbstractGridDistanceMetric { maxxy | euclidean }`

Controls how the clearance check between a pin geometry and its neighboring geometries is applied.

- `euclidean` means that the distance between two rectangles is the shortest straight line between them.
- `maxxy` means that the distance is the greater of the vertical and horizontal distance between the two rectangles.

`AbstractPinStretchOutPrb { true | false }`

Allows pins to be stretched outside the PR boundary. By default, this option is off.

`AbstractBlockageEXCEPTPGNET`

© 2020 All Rights Reserved.

This option is set for blocks that want to block routing on a given layer except for power and ground routing. This is typically true for analog or memory blocks that are sensitive to noise, and do not want to allow signal routing on the layers immediately above the block.

When the option is set, then all the blockages generated by Abstract on given metal layers will have the EXCEPTPGNET LEF attribute in the LEF file.

`AbstractTopLayerCoverBlockage {Routing Layer}`

Specifies the top metal layer for cover blockage in the layout dual view. This option is added only for Layout dual view and will not cause any conflict with the normal cover blockage option that Abstract Generator has. The already existing cover blockage options in Abstract Generator have different use model and may lead to confusion.

Note: The `AbstractTopLayerCoverBlockage` argument can only be used when the `AnnotateLayoutDualView` option is set to true.

`AbstractBlockageUserDefinedSpacing`

Allows you to specify SPACING on the blockages. The value of SPACING specified by this option will supersede the SPACING specified by Abstract Generator on the blockage. The default value for this option is `absSetBinOption("Core""AbstractBlockageUserDefinedSpacing" "")`. However, the expected value is `absSetBinOption("Core" "AbstractBlockageUserDefinedSpacing" "(M1 1.0) (M2 2.0)")`. This will specify the blockages of layer M1 with spacing 1.0 and blockages on layer M2 with spacing 2.0.

`AbstractFractureRectilinearShapes { true | false }`

Controls whether only rectilinear pin shapes are farctured.

`BlockageCutVia`

© 2020 All Rights Reserved.

When set to `true`, Abstract Generator cuts a window large enough for a via to be dropped to that pin. When set to `false`, Abstract Generator cuts a window around the pin based on the range-based spacing rule for the pin width.

`BlockageLargePurposeList`

Specifies the purposes of the big blockages.

`BlockageLargeShape`

Enables Abstract Generator to look for big blockages in the current design abstract.

`BlockageLargeShapePct`

Specifies the percentage by which the size of the big blockages needs to be increased.

Example

```
absSetBinOption "Core" "AbstractAdjustBoundaryPinsPwr" "true"
=> t
absSetBinOption "Core" "AbstractAdjustPowerGeometryGroups" "single"
=> t
absAbstract
=> 1
```

Specifies a number of power net options for cells in the `Core` bin and runs the Abstract step.

© 2020 All Rights Reserved.

absVerify

```
absVerify(
     )
     => 0 / 1
```

Description

Runs the Verify step for selected cells and creates a verify view for those cells based on the options set using `absSetBinOption`.

Arguments

None

Value Returned

`0`	Returns `0` if an error is encountered.
`1`	Returns `1` if the function executes successfully.

Associated Options

[VerifyTerminals { true | false }]
[VerifyGrid { true | false }]
[VerifyTarget { true | false }]
[VerifyTargetSelection { Silicon Ensemble | Encounter }]
[VerifyTargetSystemCMD *command_line_option*]
[VerifyTechFile *technology_file_name*]
[VerifyTargetMultiple { true | false }]
[VerifyTargetMultipleRows { true | false }]
[VerifyTargetPower { true | false }]
[VerifyTargetRouter { Nanoroute | Wroute }]
[VerifyTargetMaxRouteTime *value*]
[VerifyConfigFile *value*]
[VerifyTargetGeom *value*]
[VerifyTargetIgnore *message_identifiers*]

© 2020 All Rights Reserved.

Option Descriptions

`VerifyTerminals { true | false }`

Compares the terminals found in the logical and abstract views and reports any differences.

`VerifyGrid { true | false }`

Identifies pins and geometries that are located both on and off the manufacturing grid for each cell.

`VerifyTarget { true | false }`

Generates a test design for selected cells and runs it through the place-and-route system specified using `VerifyTargetSelection`.

`VerifyTargetSelection { Silicon Ensemble | Encounter }`

Specifies the target place-and-route system.

`VerifyTargetSystemCMD` *`command_line_option`*

Specifies the command line used to launch the target system.

`VerifyTechFile` *`technology_file_name`*

Specifies a technology LEF file to be used by the target system.

`VerifyTargetMultiple { true | false }`

Places an additional instance of the cell being verified on either side of the main instance of the cell in the test design, provided the cell's symmetry allows it. This simulates same-side cell abutment and helps identify any pins or geometry that are too close to the cell boundary.

`VerifyTargetMultipleRows { true | false }`

© 2020 All Rights Reserved.

Places cells above and below the main instance as follows.

- Flipped about the x-axis (for example, top to bottom), provided the cell's symmetry allows it
- Abutted, if the site definition for that cell has zero row spacing

`VerifyTargetPower { true | false }`

Creates a power ring around the instances in the test design and routes it in the target system.

Turning the option off means that only signal nets are routed.

`VerifyTargetRouter { Nanoroute | Wroute }`

Specifies the routing engine. The default routing engine is `Nanoroute`.

`VerifyTargetMaxRouteTime` *`value`*

Specifies in minutes the maximum amount of time that Wroute or Nanoroute spends routing the test design. If the design fails to route within a reasonable time, it might be an indication that some of pins in the abstract are difficult to access.

`VerifyConfigFile` *`value`*

Specifies the path of the configuration file for running route. Therefore, `VerifyTargetRouter` should be set to `Wroute` before using this option.

`VerifyTargetGeom` *`value`*

Specifies the `verifyGeometry` function options, which verify the geometry of the targeted abstracts. In case of geometry violations, this option creates markers corresponding to each of the violations in the Verify view. The default value is `verifyGeometry`.

© 2020 All Rights Reserved.

`VerifyTargetIgnore` *`message_identifiers`*

Specifies a list of regular expressions which are used to identify target system messages to be ignored when creating warning and error markers in the verify view

Example

```
absSetBinOption "Core" "absVerifyTerminals" "true"
=> t
absSetBinOption "Core" "absVerifyTarget" "true"
=> t
absVerify
=> 1
```

Specifies that terminals found in the logical and abstract views are to be compared, and a test design is generated for all selected cells, and then runs the Verify step.

© 2020 All Rights Reserved.

Cell Properties

Use the following functions to set and get cell properties in Abstract Generator.

- absGetCellProp
- absSetCellProp

© 2020 All Rights Reserved.

absGetCellProp

```
absGetCellProp(
     cell
     { prCellClass | symmetry }
     )
     => string
```

Description

Returns the specified property value for a cell.

Arguments

cell Specifies the name of the cell you want to query.

`{ prCellClass | symmetry }`
Specifies the propery whose value you want to know.

Value Returned

string Returns `0` if an error is encountered.

Example

```
absGetCellProp "cellABC" "prCellClass"
=> block
```

Displays the current `prCellClass` cell property value for `cellABC`.

© 2020 All Rights Reserved.

absSetCellProp

```
absSetCellProp(
     cell
     { prCellClass | symmetry }
     value
     )
     => 0 / 1
```

Description

Specifies a property value for a cell. The variables you can use for the *value* parameter depend on the property.

© 2020 All Rights Reserved.

Arguments

cell Specifies the cell whose properties are to be set.

`{ prCellClass | symmetry }`

Specifies the cell property to be edited.

value The valid values for the property `prCellClass` depend on the bin in which the cell is.

For the *Core* bin, choose from:

```
[ core | feedthru | tieHigh | tieLow | spacer |
antennaCell | preEndcap | postEndcap ]
```

For the *Block* bin, choose from:

```
[ block | cover | ring | coverBump | blackBox ]
```

For the *IO* bin, choose from:

```
[ pad | input | output | inout | power | spacer |
areaIO ]
```

For the *Corner* bin, choose from:

```
[ bottomLeftEndcap | bottomRightEndcap |
topLeftEndcap | topRightEndcap ]
```

The valid values for the property `symmetry` include:

```
[ R0 | X | Y | R90 | X Y | X R90 | Y R90 | X Y R90 ]
```

Value Returned

`0` Returns `0` if an error is encountered.

`1` Returns `1` if the function executes successfully.

Example

```
absSetCellProp "abc" "prCellClass" "core"
=> 1
```

Sets the `prCellClass` property value for the Core cell `abc` to `core`.

© 2020 All Rights Reserved.

Terminal Properties

Use the following functions to set and get terminal properties in Abstract Generator.

- absGetTerminalProp
- absSetTerminalProp

© 2020 All Rights Reserved.

absGetTerminalProp

```
absGetTerminalProp(
     cell
     terminal
     property { direction | termType | shape | netExpr | suppSensitivity |
     groundSensitivity }
     )
     => string
```

Description

Returns the value of the specified property for the specified terminal. Specify only cells that have already been processed by `absPins`. The Pins step generates the cell terminals.

Arguments

cell	Specifies the cell to query.
terminal	Specifies the terminal to query.
property { direction | termType | shape | netExpr | suppSensitivity | groundSensitivity }	Specifies the property whose value you want to know.

Value Returned

string	Returns the value of the specified terminal property.

Example

```
absGetTerminalProp "cellABC" "vdd!" "direction"
=> inout
```

Displays the current `direction` terminal property value for terminal `vdd!` in `cellABC`.

© 2020 All Rights Reserved.

absSetTerminalProp

```
absSetTerminalProp(
     cell
     terminal
     property { direction | termType | shape | netExpr | suppSensitivity |
     groundSensitivity }
     value
     )
     => 0 / 1
```

Description

Specifies a terminal value for a cell. The variables you can use for the *value* parameter depend on the *property* setting.

© 2020 All Rights Reserved.

Arguments

cell	Specifies the cell whose properties are to be set.
terminal	Specifies the terminal whose properties are to be set.
property { direction | termType | shape | netExpr | suppSensitivity | groundSensitivity }	Specifies the terminal property to be edited.
value	Specifies the value to be set for the terminal property specified through *property*. The valid values for different properties are given below. ■ direction: input, output, inout, jumper, tristate, unused ■ termType: signal, clock, analog, power, ground ■ shape: feedthru, abutment, ring ■ netExpr: In the format [@gnd:%:gnd!] ■ suppSensitivity: Specify a valid power terminal name. ■ groundSensitivity: Specify a valid ground terminal name.

Value Returned

0	Returns 0 if an error is encountered.
1	Returns 1 if the function executes successfully.

Example

```
absSetTerminalProp "cellABC" "vdd!" "direction" "input"
=> 1
```

Sets the terminal direction property value for terminal vd! in cellABC to be input.

© 2020 All Rights Reserved.

Miscellaneous

The following SKILL functions do not fit into any of the other categories.

- absDisableUpdate
- absDistributeCells
- absEnableUpdate
- absExit
- absRevalidateSelectedCells
- absSort
- absVersion
- iagCreateMenuItem

© 2020 All Rights Reserved.

absDisableUpdate

```
absDisableUpdate(
     )
     => t / nil
```

Description

Both the `absDisableUpdate` and `absEnableUpdate` functions are used to control whether or not the graphical user interface updates in response to changes in, for example, bin selection. This ensures that the GUI does not update unnecessarily when several events in a row are processed. This is inserted by Abstract Generator into the `record` file.

Arguments

None

Value Returned

`t`	Returns `t` if the function executes successfully.
`nil`	Returns `nil` if an error is encountered.

Example

```
absDisableUpdate
=> t
absDeselectBinFrom "Core" "Core"
=> t
absSelectBinFrom "Ignore" "Ignore"
=> t
absEnableUpdate
=> t
```

Specifies that the GUI is not to be updated when the `Core` bin is deselected and the `Ignore` bin selected instead.

© 2020 All Rights Reserved.

absDistributeCells

```
absDistributeCells(
     )
     => t / nil
```

Description

Distributes cells between bins according to the options set.

Arguments

None

Value Returned

`t`	Returns `t` if the function executes successfully.
`nil`	Returns `nil` if an error is encountered.

Associated Options

[DistributeBinBelowBottom *bin_name*]
[DistributeBinAboveBottom *bin_name*]
[DistributeBinAboveMiddle *bin_name*]
[DistributeBinAboveTop *bin_name*]
[DistributeHeightBottom *height*]
[DistributeHeightMiddle *height*]
[DistributeHeightTop *height*]
[DistributeCells { all | selected }]
[DistributeMethod { name | height }]

© 2020 All Rights Reserved.

Option Descriptions

`DistributeBinBelowBottom` *`bin_name`*

Specifies the bin in which to place cells that are smaller than the height set using `DistributeHeightBottom`.

The default setting is `Ignore`.

`DistributeBinAboveBottom` *`bin_name`*

Specifies the bin in which to place cells that are bigger than the height set using `DistributeHeightBottom`.

The default setting is `Core`.

`DistributeBinAboveMiddle` *`bin_name`*

Specifies the bin in which to place cells that fall within the criteria set by the `DistributeHeightMiddle` option.

The default setting is `IO`.

`DistributeBinAboveTop` *`bin_name`*

Specifies the bin in which to place cells that fall within the criteria set using `DistributeHeightTop`.

The default setting is `Block`.

`DistributeHeightBottom` *`height`*

Specifies in microns the minimum height for cells to be distributed.

Cells that are larger than or equal to the specified height but less than the specified middle height are placed in the bin set by the `DistributeBinAboveBottom` option.

The default setting is `100`.

`DistributeHeightMiddle` *`height`*

© 2020 All Rights Reserved.

Specifies in microns the middle height range for cells to be distributed.

Cells that are larger than or equal to the specified height but less than the specified top height are placed in the bin set by the `DistributeBinAboveMiddle` option.

The default setting is `180`.

`DistributeHeightTop` *`height`*

Specifies in microns the maximum height for cell distribution.

Cells larger than or equal to the specified height are placed in the bin set by the `DistributeBinAboveTop` option.

The default setting is `1000`.

`DistributeCells { all | selected }`

Specifies whether to distribute all cells or only those cells currently selected.

The default setting is `all`.

`DistributeMethod { name | height }`

Selects the method for distributing cells into bins. The default setting is `height`.

Example

```
absSetOption "DistributeMethod" "height"
=> 1
absSetOption "DistributeCells" "all"
=> 1
absSetOption "DistributeHeightTop "1000"
=> 1
absSetOption "DistributeBinAboveTop "UserBin1"
=> 1
absDistributeCells
=> t
```

Sets the distribution method to be `height` for `all` cells in the current library. Any cells of a height equal to or above `100` microns are placed in `UserBin1`.

© 2020 All Rights Reserved.

absEnableUpdate

```
absEnableUpdate(
     )
     => t / nil
```

Description

Both the `absEnableUpdate` and `absDisableUpdate` functions are used to control whether or not the graphical user interface updates in response to changes in, for example, bin selection. This ensures that the GUI does not update unnecessarily when several events in a row are processed. This is inserted by Abstract Generator into the `record` file.

Arguments

None

Value Returned

`t`	Returns `t` if the function executes successfully.
`nil`	Returns `nil` if an error is encountered.

Example

```
absDisableUpdate
=> t
absDeselectBinFrom "Core" "Core"
=> t
absSelectBinFrom "Ignore" "Ignore"
=> t
absEnableUpdate
=> t
```

Specifies that the GUI is not to be updated when the `Core` bin is deselected and the `Ignore` bin selected instead.

© 2020 All Rights Reserved.

absExit

```
absExit(
     )
     => void
```

Description

Closes Abstract Generator.

Arguments

None

Value Returned

None

Option Descriptions

`SortOrder { Ascending | Descending }`

Specifies the sort order to apply to cells. The default is `Ascending`.

`SortView { Layout | Logical | Pins | Extract | Abstract | Verify }`

Sort cells based on the presence of a named view. The default is `Abstract`.

`SortMethod { None | Name | Height | Status | Selection }`

Specifies the sort method. The default is `Name`.

Example

```
absSetOption "SortOrder" "Ascending"
=> 1
absSetOption "SortView" "Pins"
=> 1
absSetOption "SortMethod "Name"
=> 1
absSort
=> t
```


© 2020 All Rights Reserved.

Specifies that those cells in the current bin that have a `Pins` view present are sorted by `Name` in `Ascending` order.

© 2020 All Rights Reserved.

absRevalidateSelectedCells

```
absRevalidateSelectedCells(
     )
     => 0 / 1
```

Description

Revalidates all invalid views for cells selected in the Cell pane. If the view has an error or warning, this remains. The view status is no longer grayed out.

For information about the symbols denoting cell status, see Viewing Status Information.

Arguments

None

Value Returned

`0`	Returns `0` if an error is encountered.
`1`	Returns `1` if the function executes successfully.

Example

```
absSetOption "SelectName" ""cell_1"
=> 1
absSelect
=> 1
absRevalidateSelectedCells
=> 1
```

Selects a cell called `cell_1` and revalidates the views present for that cell.

© 2020 All Rights Reserved.

absSort

```
absSort(
     )
     => t / nil
```

Description

Sets sort criteria for those cells in the current bin.

Arguments

None

Value Returned

`t`	Returns `t` if the function executes successfully.
`nil`	Returns `nil` if an error is encountered.

Associated Options

[SortOrder { Ascending | Descending }]
[SortView { Layout | Logical | Pins | Extract | Abstract | Verify }]
[SortMethod { None | Name | Height | Status | Selection }]

© 2020 All Rights Reserved.

absVersion

```
absVersion(
     )
     => string
```

Description

Returns version, build, and operating system information.

Arguments

None

Value Returned

string	Returns the version information.

© 2020 All Rights Reserved.

iagCreateMenuItem

```
iagCreateMenuItem(
     )
     => t / nil
```

Description

Creates a custom menu item with the user-specified name and adds it to the integrated Abstract Generator. The menu is created if it is not already available, and the menu item is added under it. Function name is the callback procedure for the menu item.

Arguments

menuName	Specifies the name of the menu under which the menu item is created. For example, Tools, Custom_Hook, etc.
menuItemName	Specifies the name of the menu item. For example, Save, Open, etc.
functionName	Specifies the name of the function that can be run using the menu item. *functionName* can be passed as a string or a symbol.

Value Returned

`t`	Returns `t` if the menu item is created.
`nil`	Returns `nil` if no menu item is created.

Example

```
iagCreateMenuItem("Custom_Menu" "Open Pre_Hook" "displayForm")
```

Creates a menu item named `Open Pre_Hook` under the menu `Custom_Menu` and registers the procedure call `displayForm` on clicking the menu item.

© 2020 All Rights Reserved.

iagGenAbstract

```
iagGenAbstract(
     t_libName
     l_cellNames
     t_viewName
     t_binName
     t_optionsFilePath
     )
     => t / nil
```

Description

Runs Integrated Abstract Generator on the specified cellviews with the given settings.

Arguments

t_libName	Name of the library in which the cellviews reside.
l_cellNames	List of cells for which abstracts are to be generated.
t_viewName	View name for which abstracts are to be generated.
t_binName	Bin in which the cells are to be placed. Valid values: `"Core"`, `"Block"`, `"IO"`, and `"Corner"`
t_optionsFilePath	Absolute path to an options file that defines the abstract generation rules.

Value Returned

`t`	The abstracts were generated.
`nil`	The abstracts could not be generated because of an error while running the tool. The error message is displayed separately.

Example

```
iagGenAbstract("test" list("merge1" "merge2") "layout_merge" "Core" "/home/user2/
.abstract.options")
```

Generates abstracts for the `layout_merge` view of cells `merge1` and `merge2` in the `test` library. The settings in the specified options file are applied and the cells are placed in the `Core` bin.

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

C

Library Design Guidelines

This appendix contains guidelines that will help you optimize your library to reduce die size and maximize the effectiveness of the place-and-route software.

A properly designed cell library is one of the most important factors in obtaining optimum results from Innovus™ Digital Implementation System (Innovus). A poorly designed library can lead to increased run times, undesirable placements, poor routing, and ultimately, less than ideal performance.

Although the following suggestions and guidelines are written with Innovus in mind, many also apply to any advanced place-and-route tool. It might be difficult, if not impossible, to incorporate all of the following guidelines into your library: the goal, however, is to adhere to as many of these features as possible in your library design.

Before detailing the library development guidelines, it is important for the designer of the library to keep in mind that minimizing cell sizes does not necessarily minimize the total die size. In fact, some of the suggestions in the following pages will cause an increase in the size of the individual cells, but will decrease the overall die size due to reductions in routing among the cells.

This appendix does not detail the procedures for creating or validating your library. The main sections covered in this appendix are:

- General Guidelines
  - General Rules for All Cell Types
  - Designing Pins for Cells
  - Designing Standard Cells
  - Designing IO and Corner Cells
  - Designing Blocks

© 2020 All Rights Reserved.

- Determining Routing Grid Pitch
  - General Guidelines for Setting Pitch
  - Setting Pitch for Layers with the Same Preferred Direction
  - Setting Pitch for Via Stacking
  - Designing a Gridded Library for Innovus
  - Converting an Existing Library
- Modeling Pins
  - Deciding Pin Layers
  - Power to Blockage Violations
  - Virtual Tracks
  - Off-grid routing on the virtual track might cause violations with fractured pin or blockage rectangles.
  - Centering On-grid Pins
  - Pins with Potential Routing Problems
- Blockages
  - Blockages and Wiring Density
  - Blockages and Pin Accessibility
  - On-grid Versus Off-Grid Blockages
- Cell Porosity and High Density
  - Routing Pitch
  - Pin Spacing
  - Pin Staggering
  - Defining Tracks for Wide Wire Designs
  - Variable-Height Versus Multiple-Height Cells
  - Non-rectangular Blocks

© 2020 All Rights Reserved.

General Guidelines

This section describes the models the software uses and gives you some general tips on the best ways to model the cells in your library.

General Rules for All Cell Types

- Every logical pin must have a physical description that contains at least one port. Each port must have at least one physical geometry.
- All objects belonging to the cell must be inside the cell boundary.
- In some technologies, power and ground pins extend to the edge or outside the boundary.
- The edges of pins and blockages must be at least one half the minimum spacing away from the cell boundary.
- Minimizing cell size does not necessarily minimize die size. It is better to increase the cell size slightly to optimize routing than to minimize cell size without keeping routing considerations in mind.

Designing Standard Cells

This section provides suggestions for optimizing your design at the library level or inter-cell level, to ensure that the individual cells work together as well as possible.

Optimize Cell Widths

- Avoid creating standard cells of the same type that vary greatly in width.

  This results in unbalanced row lengths. In general, the width of the largest standard cells should be no more than five or six times the width of smallest standard cell of the same type in your library.

  If you need more complex cells, it is better to double or triple the height of the cells and keep a reasonable cell width. Innovus can optimize the area utilization by mixing single- and multiple-height cells in the core area without creating gaps around multiple-height cells.

- All cell widths should be a multiple of the pin grid.

  If a cell width is not a multiple of the pin grid, even though its pins are on the grid, the cell will cause the cells adjacent to it to be misaligned with respect to the grid. This makes

© 2020 All Rights Reserved.

having pins on grid useless because the misalignment of the other cells causes their pins to be off-grid, thereby using more horizontal or vertical tracks than needed.

Making the cell width a multiple of the pin grid can increase the size of the cell, but will yield a reduction in the total area of the die.

- Placement should also be done on a placement snap grid so that the edges of the cells are also on the grid.

Optimize Cell Heights

Optimal cell heights vary depending on the row style your design uses.

Of the four styles described below, using the second style is highly discouraged. You can use any of the other three for Innovus. The last two row styles typically result in the smallest die sizes.

- **Style 1:** Standard height cells in single cell rows

Typically, these designs have two or more layers. Cell height is limited. You can have power sharing if you design it in. Power sharing can reduce die size by reducing the number of channels.

© 2020 All Rights Reserved.

- **Style 2:** Variable height cells, in single cell rows

These designs are usually two-layer designs. They are not efficient for three or more layers. They allow cell heights to vary by about 30%, but do not allow for power sharing.

© 2020 All Rights Reserved.

- **Style 3:** Single-height and double-height cells, in double cell rows

These designs have two or more layers. Single-height or double-height cells are arranged in double rows. This kind of design is meant for power sharing.

- **Style 4:** Multi-height cells in a sea-of-cells design

These designs feature three or more layers. They do not allow two-layer routing and have no channels, so you cannot use a channel-based router. You can have single-, double-, or multi-height cells. This kind of design permits the most efficient power sharing of both power lines.

Optimize Porosity

- Medium and large cells should have the same porosity, primarily in *M2*.

  This means that the percentage of free *M2* tracks should be the same for all the medium and large cells in the library. If the porosity is not uniform, the router might have trouble finishing if cells with low porosity are placed near each other.

  For small cells (one to three tracks wide) you can let the placer optimize porosity by spacing them apart where necessary. However, careful design can improve routability significantly.

© 2020 All Rights Reserved.

For example, you can maximize *M1* left/right access to pins on small cells, as shown in the following figure. This optimizes *M2* porosity by maximizing the amount of routing done on *M1*. Small cells often have this kind of routing.

You can also design small cells to have multiple accessible pins or open grids so that you can get Z-shape *M2* routes where possible, as shown in the following example.

© 2020 All Rights Reserved.

Designing Pins for Cells

This section contains guidelines for designing pins so that you maximize the routability of the cells and improve the run time and memory performance of the router. The more porous the design, the easier it is for the router to make connections.

Placing Pins on the Grid

Every pin should cover at least one grid intersection because the software connects most efficiently to pins at grid intersections. If you have off-grid pins, the software creates a pseudogrid through the center of each off-grid pin.

Use the following guidelines to optimize pin placement:

- Place pins on-grid wherever possible.
- Ensure that off-grid pins follow spacing rules by using the pseudogrid point at the center of the pin geometry.
- Center the pin via on the routing grid for its layer.
- Place special pins in such a way so that they do not obstruct routing on an adjacent track. You can place special pins off-grid for this.

Innovus has gridless routers but they handle gridded pins better than off-grid pins. A pin is considered to be on-grid if it meets the following criteria:

- The pin rectangle is centered at the grid intersection.
- The pin rectangle encloses the grid intersection point on all four sides by half the minimum metal width of that pin shape layer. The grid intersection is from the x and y routing grid pitch for that layer. The router routes to the pin rectangle only at that grid intersection.
- The router only routes to the center of the off-grid pin port rectangles if this is legal.

© 2020 All Rights Reserved.

The following figure shows a LEF standard cell with shapes similar to layout data.

Pin A has several off-grid shapes and one on-grid shape. Pin B has two off-grid polysilicon pin shapes. Pin O has one on-grid shape. The off-grid shapes slow down the router and lead to inefficient routing. Further, if the spacing rules for this example did not allow *M1*/*M2* vias to be placed on pin A, the router would not be able to connect to Pin A. This could result in an inefficiently routed design.

Off-grid pins can affect your design in a number of ways. Some of these are listed below:

- Large die size

  The router creates jogs to compensate for pins that are not aligned. This forces the channels to expand vertically and increases the die size.

© 2020 All Rights Reserved.

- Overconstrained design

  The more constraints there are, the harder it is for the router. The following figure shows how one off-grid pin increases the number of constraints.

On-grid pins: three pins, two constraints

A constrains C.
C constrains A.
B has no constraints.

Off-grid pin: three pins, four constraints

A constrains C.
C constrains A.
B constrains C.
C constrains B.

- Longer run-times

  This design slows down the router significantly because the router is more constrained. Each off-grid track can interfere with adjacent tracks.

- Optimistic global router results

  The global router might not recognize off-grid pins as blocking the routing track and, as a result gives you optimistic results.

Handling Special Pins

Special pins are pins routed by the special net router instead of the final router. Special pins have different shapes, depending on their function. On cells, you can have feedthrough or abutment special pins.

- Non-interference with adjacent grid lines.
- Define special pins in the `SPECIAL NETS` section of the DEF file.

© 2020 All Rights Reserved.

- Place all power and ground feedthrough and abutment pins at the same axis on opposite sides of the cell.
- Position the power and ground feedthrough or abutment pins so that they align with corresponding pins in other cells.
- Avoid creating power and ground pins in *M2*, because *M2* within the cell makes it harder to route over the cell.
- Define a *shape* property for power and ground pins with the value set to either *feedthru* or *abutment*.
- Set the same y offset for power and ground pins from the cell origin so that the *FollowPins* command works properly. You can specify the origin in the LEF file or use the default, which is the lower left corner.

Maximizing Routing Resources

The more routing resources you have available, the easier it is for the router to connect. You must maximize the resources in your most constrained layer. In three-layer designs where *M2* is vertical and *M1* and *M3* are horizontal, *M2* is typically the most constrained layer.

The following are some suggestions:

- Maximize pin accessibility.

  Make cell pins as accessible as they can be by declaring the complete geometry of the pin as a pin. This will allow the router to connect to the best location on the pin as it sees fit, rather than forcing it to connect to a particular point.

  Also, avoid having geometries on the same layer close to and around the pin, in effect, blocking it. This makes the router unable to make a connection to the pin.

  Special care should be taken not to block access to pins through vias. For example, avoid *M2* geometries too close to a *M1* pin such that *M1* to *M2* via cannot be dropped in order to access the pin on *M2*.

© 2020 All Rights Reserved.

- Stagger pins in the x and y directions.

  Staggering pins provides additional vertical routing resources, as shown in the following figure:

  If two or more pins are aligned vertically, then the vertical track going through the pins can become blocked, forcing some of the horizontal routing resources to be used to connect to the pins.

  By staggering the pins in the vertical direction, especially at a pitch equal to integer multiples of the pitch of the vertical routing track, a higher degree of flexibility is given to the tool to connect to the pins in either direction.

- Minimize the number of geometries within the cell that are on the constrained layer.

  Any geometry on the constrained layer within the cell becomes a blockage to routing in that layer. For three-layer designs where *M2* is the vertical layer, you can maximize routing resources by creating internal cell geometries in *M1* and polysilicon, and leave *M2* and *M3* for routing.

- Allow extra routing resources in the constrained layer.

  In libraries for gate isolation base arrays that might be congested in *M2*, you can add extra *M2* routing resources as padding. For example, a two-input NAND gate might be three tracks wide with every track obstructed by a pin. You can model this as at least four tracks wide to provide an extra track for routing.

- Do not add pieces of metal to provide extra connection points.

  In some libraries, you add a piece of *M2* to a pin to provide the router with an extra connection point if *M1* is blocked. This is counter productive because unused *M2* areas become blockages that limit the flexibility of the router. You do not have to add extra metal, because the router can drop vias to the pins.

© 2020 All Rights Reserved.

Designing IO and Corner Cells

Abstract Generator uses the cover blockage model for IO cells, preventing the router from routing over IO cells.

Abstract Generator uses the same model for generating abstracts for corner cells.

Designing Blocks

- Maximize porosity for over-the-block routing.

  Leave unobstructed areas for over-the-block routing. Minimize the number of M2 and M3 geometries inside the block, so that these routing resources can be used to route over the block.

- Create power rings around large macro blocks.

  It might be easier to accommodate power routing around a macro block if you create a power ring around it and connect the internal buses to the ring.

- Pad around block boundaries.

  It is always good to have extra routing resources around a block, because this area can get congested. Leave extra room around the corners.

- Create as much open vertical *M2* space as possible on the left and right sides.
- Create as much open *M1* and *M3* space as possible on the top and bottom.
- Space pins apart by at least line-to-via spacing.

  If your design requires line-to-line spacing, add flare outs to line-to-via spacing even if this sticks out from the macro, as shown in the following figure.

© 2020 All Rights Reserved.

Determining Routing Grid Pitch

This section discusses the various guidelines associated with determining routing grid pitches and recommended pitch setting strategies.

General Guidelines for Setting Pitch

You must define horizontal and vertical grids for routing on every routing layer. To do this, you must do the following:

- Select the preferred and non preferred routing direction for each layer.

  One common convention is *HVH*, where the preferred direction for routing *M1* and *M3* is horizontal and the preferred direction for *M2* is vertical.

- Define a minimum pitch for each layer in the preferred direction track for that layer.

  This pitch must be at least line-to-via distance. Anything less will result in low utilization rates and longer run times, because the router will not be able to place vias next to wires on adjacent tracks.

  The following figure shows line and via definitions for track pitch:

© 2020 All Rights Reserved.

The most common track pitch is line-to-via, because it is the minimum pitch in which you can place a via next to a route. It is calculated as follows:

half the metal width for the layer + the metal spacing + half the via.

- Assign tracks in the non-preferred direction for every routing layer.

  Align these tracks with preferred direction tracks on adjacent layers, even if this is different from the track pitch of that layer in the preferred direction. Lining up the tracks lets the software create jogs at locations where vias can be placed or where pins can be connected to consistent xy locations.

- Allow for diagonally adjacent vias.

  Increase spacing to allow vias to be placed diagonally adjacent to each other. This might make the orthogonal spacing slightly more than line-to-via, but will reduce overall die

© 2020 All Rights Reserved.

size, because lack of via resources causes more congestion and longer routes to route around congested areas.

- Maximize total routing track resources

  If you have an HVH layer scheme, keep *M1* and *M3* spacing to the minimum, and make any needed spacing increases in M2. Minimizing spacing on two layers (*M1* and *M3*) gives you more total routing resources than minimizing spacing on one layer by the same amount.

  To a certain extent, the placer also adjusts for different horizontal and vertical demands on the routing resources. You can also use a rectangular die instead of a square one to maximize routing resources.

- Use rectangular vias if that allows you to have smaller track pitch and more total routing resources overall.

  Ensure that the rectangular via and pitch still allows diagonal vias.

  Normally, you maximize your total routing track resources. However, if one direction is constrained, you might want to maximize your resources in that direction.

  For example, take an HVH library with the same line-to-via spacing for all three layers and no diagonal vias. To allow for diagonal vias, increase the spacing for the horizontal layers (*M1* and *M3*) to maximize the vertical routing resources on *M2*. This might not optimize the overall routing resources, but it reduces the congestion on the critically constrained *M2* layer.

© 2020 All Rights Reserved.

Alternatively, if the cells have no free *M1* tracks and the *M3* pitch is much larger than *M2*, you can increase the spacing on *M2*.

Another technique for handling an overconstrained layer is to use a die aspect ratio that minimizes the routing demand for that layer.

Setting Pitch for Layers with the Same Preferred Direction

- Align the tracks on the layers as much as possible
- If you cannot align the tracks, keep the pitch ratio between them simple, for example, 1:2 or 2:3.

  The following figure shows a ratio of 2:1 between *M1* and *M3*. The figure shows preferred direction tracks. Non-preferred direction tracks should be the same.

If *M1* and *M3* have complex ratios, such as 11:8, you can run into the following problems:

- If you have inter-layer spacing rules between cut layers, vias will often be limited on two tracks instead of one.

© 2020 All Rights Reserved.

In the following figure, *V12* prevents the router from dropping *V23* vias above and below it on the *M3* tracks. If the *M1* and *M3* tracks were aligned, the via (*V12*) would block only one *V23* via.

- This design can seriously affect memory allocation at run time. With a ratio of 11:8, you must create a non preferred *M2* track for every *M1* and *M3* track. Because these tracks align only at every eighth track, the result is more *M2* tracks than required with a 1:1 ratio, and this affects memory allocation.
- With a ratio such as11:8, your choice of cell heights is limited. This is because the cell height should be a multiple of the common denominator of the track spacing.
- If your technology allows center-stacked vias, a ratio like 11:8 will produce few.

Setting Pitch for Via Stacking

- If two metal layers have the same routing direction, make the relationship one of the following: 1:1, 1:2, 2:3, or 3:4.

  Vias cannot be stacked where *M1* and *M3* are not coincident. You can only stack vias at grid intersection points where all layers in the stack exist. If the relationship is not one of the above, it is difficult or impossible to place vias.

© 2020 All Rights Reserved.

If your process allows stacking of vias, you *must* use one of the above ratios. To stack vias everywhere, use 1:1. This is described in the following figure.

Designing a Gridded Library for Innovus

Use the following general guidelines to simplify the internal data structures that Innovus uses during routing (assuming horizontal rows):

- Make cell width a multiple of the vertical routing track pitch.

  Cell widths should be a multiple of the vertical routing pitch if there is only one vertical routing layer.

- Make cell heights a multiple of the horizontal routing layer pitch.

  If there is more than one horizontal routing layer, use a multiple of the greatest common denominator of the horizontal track spacing.

- Create pins that cover routing grid intersection points by half the minimum width.

  The CPU time increase for on-grid versus off-grid routing varies from 20% to 30%, depending on how many off-grid pins are in the library. The quality of route is not significantly degraded with off-grid routing.

- Make blockages cover grid intersection points.

  Blockages and pins should cover grid intersection points even after the blockages are fractured.

© 2020 All Rights Reserved.

The routers can use any grid intersection point not covered by a pin or blockage shape.

Converting an Existing Library

The following guidelines apply if you have to create a routing grid for a library that was not designed with a routing grid in mind, or for a library that has a routing grid designed for an older technology that needs to be reduced.

- Calculate all minimum line-to-via and via-to-via pitches for all the layers. Calculate minimum xy distances for all diagonal vias.
- Determine the routing pitch and offset for each layer by examining some cell layouts from the existing library.

  You can get the *M2* preferred pitch (x coordinate) from the pin center-to-center distance of layout shapes. Getting the *M1* pitch (y coordinate) might be harder because many cells in older libraries were not designed with a y pitch in mind.
- If the designed x or y pitch is larger than the line-to-via minimum, reduce the pitch.

  It might be easier to reduce the y pitch than the x pitch. Reset cell boundaries to be a multiple of the new pitch.
- Define different *M1* and *M3* pitches to maximize routing resources.

  Remember to use simple ratios. The cell boundary must be a multiple of both layers.
- Use rectangular vias for routing if that allows the grid you want.

© 2020 All Rights Reserved.

Modeling Pins

Deciding Pin Layers

- Avoid creating *M2* geometry on pins. This blocks routing for other nets if the *M2* geometry is not required.
- Create an *M2* pin only if the pin is accessible only by *M3*. That is, it is blocked completely with *M1* and *M2*.

Power to Blockage Violations

A common library problem occurs during the power and ground connections by the special router used in Innovus.

If the blockage does not keep a legal clearance to the power and ground pins, as shown in Figures (a) and (b) below, false violations are highlighted that make it difficult to identify real violations.

(a) Abutted blockage and pwr/gnd pin

(b) Disjoint blockage and power pin d< min. spacing

(c) Shrink the blockage d>= min.spacing

© 2020 All Rights Reserved.

Virtual Tracks

Each off-grid rectangle must be made on-grid in order for Innovus to recognize it. This is accomplished using virtual tracks.

Certain libraries require the creation of thousands of virtual tracks because of the presence of off-grid geometries, which must be avoided during routing. The following figure shows virtual tracks in use.

Off-grid routing on the virtual track might cause violations with fractured pin or blockage rectangles.

When virtual tracks are present, the router will use either virtual tracks or regular tracks, depending on which are closest.

© 2020 All Rights Reserved.

Centering On-grid Pins

If off-grid pin access is not allowed, you should center pins on grid intersection points. If a pin shape touches a grid intersection point, it is considered on-grid.

If your pins touch grid points, and you do not allow off-grid routing;

- The router will recognize it as needing to be routed, and will avoid it when routing other nets.
- Depending on the pin shape and position, the router might be unable to route to the pin, or make an efficient connection.

Routing problems can be caused by incorrectly defined on-grid pins. These problems are sometimes referred to as off-grid problems. The router and verify geometry (in Innovus) do not consider a notch a violation. These violations are found by the `samenet` check.

© 2020 All Rights Reserved.

Pins with Potential Routing Problems

- The pin geometry interferes with the adjacent grid without covering it.

  The problem with off-grid pins that do not cover grid points but are too close to allow routing, is that the global router will not see the track as obstructed. The global routing scores will not consider that the track is blocked by the off-grid pin and will give you over-optimistic results.

- Vias (in pins) are not centered on the routing grids for its layers.

  If off-grid pins exist, and the router has been set to allow off-grid pins, then grids are created by the router. Any pin that does not cover the routing grid with enough area to place a via is considered off-grid, and a track is created through the center of the pins. Off-grid tracks will slow the router by requiring more memory. Routing is slower because wires on each off-grid track interfere with adjacent on-grid routing.

- Special pins need not be on-grid, but should follow blockage rules.

  The following figure shows potential routing problems.

© 2020 All Rights Reserved.

Blockages

All geometries on routing and cut layers that are not associated with a pin are identified as blockages by Abstract Generator. A blockage is a keep-out area for any routing layer or cut layer.

You can use Abstract Generator to create blockages. You can specify the *Geometry Specification* for the layers on which the blockage is to be created. Alternatively, from Abstract Generator you can open any one of the layout, pin, extract, or abstract cellview in VLS L and edit them to add blockages. For information about adding blockages by using VLS L, see "Creating Blockages" in the *Virtuoso Layout Suite L User Guide*.

Blockages and Wiring Density

Use the following guidelines when blocking a routing layer:

- If all of the routing is blocked for a layer, then create a single blockage over the area instead of many separate blockages.

  This simplifies the LEF file generated for the abstract.

- When drawing a blockage over the whole cell, leave half a minimum spacing between the blockage and the boundary.

- If there are free tracks available, leave them open in the abstract.

  This can increase wiring density.

© 2020 All Rights Reserved.

In the following figure, the blockages in the cell on the right allow *M1* routing to the pins.

© 2020
All Rights Reserved.

Blockages and Pin Accessibility

Pins are accessible from the side if they are less than one routing grid distance from the blockage edge, and the command `FROUTE.ALLOW.PINACCESS.ATBLOCKAGEEDGE` is set to `TRUE`.

Note: The command `FROUTE.ALLOW.PINACCESS.ATBLOCKAGEEDGE` controls *FRoute* only, and not the line probe router.

The line probe router (LPR) will *not* route to any pin that is surrounded by a blockage on that layer.

In the following figure, pins are accessible by routing over the cell with *M2*, and dropping a via to *M1* (Via12 is a via from *M1* to *M2*).

If your library has blockages defined as shown above, you must route the clock tree with *GRoute* and *FRoute*, and not *CRoute* Balance.

© 2020 All Rights Reserved.

On-grid Versus Off-Grid Blockages

On-grid blockages completely cover the grid points that are interfered with.

Slivers that result from fracturing can cause edge-to-edge spacing violations. This happens when a sliver is off-grid, as shown on the right of the figure below. The sliver covers no grid intersection points, but interferes with routing.

The blockage on the left covers three grid points and effects only one vertical routing track. This is an on-grid blockage. It allows a via to be placed without violations. The blockages on the right cover three grid points and interfere with two vertical routing tracks. The sliver also causes an edge violation with the via. The sliver is an off-grid blockage.

Libraries with no off-grid blockages allow the best routing density. The final router runs faster if all blockages within cells are on-grid.

© 2020 All Rights Reserved.

Cell Porosity and High Density

Cells are more porous if there is free routing resource to route over the cell. A porous library allows high core utilizations.

Routing Pitch

The pitch you choose affects the routing density. The pitch is usually a value between via-to-via spacing and line-to-via spacing.

Via-to-Via Spacing

This is the spacing required for adjacent vias and is illustrated in the figure below.

© 2020
All Rights Reserved.

Line-to-Via Spacing

The line-to-via dimension is the distance that the routing pitch requires in order to legally place a via on the track next to a wire (line). This is illustrated in the figure below.

© 2020
All Rights Reserved.

Adjusting the Routing Pitch

Most processes allow diagonal vias when the routing pitch is set to line-to-via. In some cases it is necessary to adjust the vertical routing pitch to allow diagonal vias. This adjustment is referred to as *expanded line-to-via spacing*.

Expanded line-to-via pin spacing gives better routing density than via-to-via spacing. Use via-to-via spacing only if your process requires large vias. This is illustrated in the figure below.

If diagonal vias are not allowed when using the line-to-via routing pitch, expand the routing pitch just enough to make diagonal vias allowed. The amount that the pitch must be expanded depends on the cap size of the vias.

When diagonal vias are placed, the spacing (d) must be greater than the *M1*-to-*M1*, *M2*-to-*M2*, and *M3*-to-*M3* spacing rule, depending on the layers in the via.

© 2020 All Rights Reserved.

Pin Spacing

Pin spacing is usually more than line-to-via spacing and less than via-to-via spacing. Start with line-to-via spacing and, if necessary, increase the pin spacing to allow access to both pins by dropping vias (adjacent vias).

Normally, the maximum spacing necessary is via-to-via spacing.

Single Grid Pin Spacing

A single grid pin is a pin with only one port that covers only one grid point.

Use pin spacing so that it is legal to drop vias on all pins, as shown in the following figure.

If you place single grid pins on orthogonally adjacent grid points, you *must* use via-to-via routing pitch; otherwise, the router might not be able to access the pins.

If you place single grid pins on diagonally adjacent grid points, your routing pitch *must* be able to accommodate diagonal vias.

© 2020 All Rights Reserved.

Pin Staggering

If you have one vertical routing layer, and line-to-via pin spacing, you should stagger the pin placement, as shown in the following figure.

If pins are at the same y coordinate, there is no legal way to route through the area. However, placing pins at alternate y coordinates allows the final router to route vertically through the cell. (Allow one vertical feedthrough for every three pins.)

If pins are not staggered as shown, the router can only route vertically between cells. This makes designs with high row utilizations unroutable.

© 2020 All Rights Reserved.

Defining Tracks for Wide Wire Designs

If you route wide and narrow wires, you can get better on-grid routing density by doubling the number of tracks, as shown in the following figure.

For designs with a combination of wide wires and narrow wires,

- The design on the left shows the closest that wide wires can be placed using the normal, expanded line-to-via routing pitches.
- The design on the right shows better routing density, if you double the number of tracks.

© 2020 All Rights Reserved.

Variable-Height Versus Multiple-Height Cells

Cell heights in your library affect floorplanning and placement. This is shown in the following figure.

Variable-height cells
(Cell heights are not multiples of each other.)

Multiple-height cells
(Single, double, triple, etc.)

For more information about variable-height cells see Variable Height Cells and for more information about multiple-height cells see Multiple Height Cells.

Variable Height Cells

If your design has variable height cells, define only one site type. The following guidelines should be used to assist you.

- The tallest cell's height should be used to define row (site) height.
- Cells are aligned by the lower left corner.
- Power must abut between all cells in the row.
- Placement space is wasted where short cells are placed.
- Floorplanning the core area is simplified because there is only one type of core row.

© 2020 All Rights Reserved.

In the following figure, the cell heights are not multiples of each other.

As a result of this design,

- Floorplanning is simplified because additional row types are not added.
- Placement space is wasted because the rows do not overlap.
- The lower left corner of the non rotated cell's place-and-route boundary is used as the alignment point.

If cell heights are multiples of each other (single and double height cells, for example), floorplanning is handled differently.

Using the Overlap Layer

You can use the overlap layer to increase placement density and eliminate most of the empty space. The following guidelines should be used to assist you.

- The site for each cell is the same height (use the tallest cell to define the site).
- The overlap layer defines the cell height.
- Empty space above short cells is used by taller cells in the next row.

© 2020 All Rights Reserved.

- The placer will resolve overlaps as shown.

If power overlaps:

- The cells must have power pins defined so they connect when cells abut.
- If flipping of rows is allowed, you must define power to accommodate this.

Cell rows can overlap where power does not overlap.

Multiple Height Cells

The following guidelines should be used to assist you when using multiple-height cells.

- These cells have a height that is a multiple of the normal cell height.
- Define a site for each cell height.
- Rows can overlap as long as the power rails align.
- A true channel-less library, where no routing channels are needed between core cell rows, provides best placement.
- Advantages
  - Multiple-height cells utilize placement space better than variable-height cells.
- Disadvantages
  - Overlapping single and multiple-height rows complicates floorplanning.
  - Utilization calculations are more complex.

If a cell is over twice as high, then it limits routing space between the short rows, based on the large cells abutment.

Currently, Innovus does not report utilization values that take into account overlapping rows.

© 2020 All Rights Reserved.

Non-rectangular Blocks

If you have non-rectangular blocks, some of the area might be available for standard cell placement.

- In the LEF file use the overlap layer to describe the area that is not available for placement of other cells.
- Otherwise, by default, the bounding box of the boundary or the `SIZE` statement in the LEF file is used for the shape of the cell.
- Use Abstract Generator to translate the boundary layer into the LEF overlap layer.

Define an blockage on the `OVERLAP` layer in the block (megacell) over the areas that are not available for cell placement.

© 2020 All Rights Reserved.

Summary

For an optimal design, follow these guidelines:

- Determine the routing grid for each layer.
- Try to keep pin and blockages on routing grids.
- Ensure that pins will be accessible after all connections are made.
- Keep cell size a multiple of the greatest common denominator of grids in each direction.
- Create a site for each cell height and multiple of grid minimum.
- Create place-and-route boundaries around cells.

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

D

Using Regular Expressions in Abstract Generator

This appendix contains guidelines on how to use regular expressions in Abstract Generator. This information is useful if you use regular expressions to complete particular entry fields in the Abstract Generator GUI.

Note: Abstract Generator does not check the entered regular expressions against any syntax rules.

Regular Expression Syntax as Used in Abstract Generator

Regular expression are strings of characters used for pattern matching. The characters in a regular expression are classified as either special or ordinary.

The table below lists the special characters and describes how to use them. Any character in a regular expression that is not special is considered ordinary, unless preceded by the character \ (See Special Constructs for "\" Supported by Abstract Generator). You can concatenate any two regular expressions to produce a new regular expression. For example, concatenating the regular expressions `f` and `i` produces a regular expression that matches the string `fi`. By incorporating special characters, you can create more sophisticated regular expressions that matches variety of strings.

© 2020 All Rights Reserved.

Special Characters Supported by Abstract Generator

A description of each special character and its intended use is detailed below:

Special Character	Description of Use
`.` (period)	This is used as a special character to match any single character except a new line. If you use concatenation, you can construct regular expressions such as `x.y`, which will match any three character string beginning with an `x` and ending in a `y`.
`*`	This special character is not a construct in itself; it is a suffix, which means that the preceding regular expression should be repeated as many times as possible. For example, in `go*`, the `*` applies to the `o`; so `go*` would match one `g` followed by any number of `o`s. Thus, `go` has a repeating `o`, but not a repeating `go`. The string matcher processes a `*` construct by matching as many repetitions that can be found. It will then continue with the rest of the pattern entered. If this fails to find a match, it backtracks, discards some of the matches of the `*` construct, and again tries to match the rest of the pattern. For example, matching `ca*ar` against the string `caaar`, the `a*` first tries to match all three `a`s, but the rest of the pattern is `ar` and there is only `r` left to match, so this attempt fails. The next alternative match is for `a*` to match only two `a`s. With this last choice, the regular expression matches successfully.
`+`	This is another suffix character, similar to `*`, except that it requires that the preceding expression be matched at least once. For example, `ca+r` will match the string `car` and `caaar`, but not the string `cr`, whereas `ca*r` would match all three strings.
`?`	This is another suffix character similar to `*`, except that it can match a preceding expression once or not at all. For example, `ca?r` will match `car` or `cr`, but nothing else.

© 2020 All Rights Reserved.

`[ ... ]`

The `[` begins a *character set*, which is terminated by the `]`. Using its simplest case, the characters between the two form the character set. Therefore, `[ad]` matches either one `a` or one `d` and `[ad]*` matches any string composed of just `a`s and `d`s.

You can also specify *character ranges* within a character set. This is done by inserting a hyphen (`-`) between the first and last characters in the range. For example, `[a-z]` will match any lower case letter. You can also use ranges in conjunction with any individual characters, for example `[a-z$*]`, which will match any lower case letter or a `$` or a `*`.

Note: Any special characters contained inside a character set are treated as ordinary. A new set of special character exists inside character sets, specifically `]`, `-`, and `^`.

If you want to include `]` as a special character in the character set, ensure it is the first character listed in the character set. For example, `[]a]` matches `]` or `a`.

If you want to include `-`, you must specify this as `---`, which is a range containing only `-`.

If you want to include `^`, you must make it any other character than the first character in the set. See `[^ ... ]` below for further information.

`[^ ...]`

Using `^` at the beginning of a character set begins a *complement character set*. This allows you to match any characters except the ones specified. For example, [`^a-z0-9A-Z]` would match all characters except letters and digits.

The `^` is only a special character if it is the first digit in a character set.

`^`

When not part of a character set `^` is a special character that matches an empty string, but only if at the beginning of a line in the text being matched.

For example, `^foo` matches a `foo` that occurs at the beginning of a line.

© 2020 All Rights Reserved.

`\`	The `\` special character has two functions; it quotes special characters (including `\`), and secondly it introduces additional special constructs. Because the `\` quotes special characters, `\$` is a regular expression that matches only `$`, and `\[` is a regular expression that matches only `[`, and so on. See Special Constructs for "\" Supported by Abstract Generator for more information about the use of this character.
`$`	The placement of the `$` special character at the end of an entire regular expression constrains that regular expression to match a final segment of a line.

Examples

Regular expressions can be used when completing the names of the pins on the Map Tab of the Pins step. (see Specifying Pin Names)

Example regular expression entries in the *Power pin names* field could include:

- `^ovdd$` which matches the text `ovdd`
- `^ovdd` which matches any text that begins with `ovdd`
- `ovdd` which matches any text containing `ovdd`
- `vdd*` which matches `vdd`, `vddd`, `vddd`,...
- `vdd.*` which matches `vdd` or `vddfred`

For more examples, see Writing Regular Expressions.

Special Constructs for "\" Supported by Abstract Generator

In general, `\` followed by any character matches only that character. There are, however, several exceptions to this rule, where characters preceded by `\` are special constructs. Such characters are always ordinary when encountered on their own.

The following table describes possible `\` special constructs are as follows:

`\` **Special Construct**	**Description of Use**

© 2020 All Rights Reserved.

`\|`

This specifies an alternative.

For example, `foo\|bar` matches either `foo` or `bar` but no other string.

Full backtracking capability allows you to handle multiple uses of `\|`.

`\( ... \)`

This grouping construct serves three purposes:

1. To enclose a set of `\|` alternatives for other operations. Therefore, `\(foo\|bar\)x` matches either `foox` or `barx`.
2. To enclose a complicated expression for the postfix `*` to operate on. Therefore, `ba\(na\)*` matches 'bananana' with any (zero or more) number of `na` strings
3. To mark a matched substring for future reference. See `\n` below.

`\n`

After the end of a `\( ... \)` construct, the matcher remembers the beginning and end of the text matched by that construct. Then, later on in the expression, you can use `\` followed by *n* to mean "match the same text matched the *n*th time by the `\( ...\)` construct."

`\'`

This matches the empty string, provided it is at the beginning of the buffer.

`\'`

This matches the empty string, provided it is at the end of the buffer.

`\B`

This matches the empty string, provided it is not at the beginning or end of a word.

`\<`

This matches the empty string, provided it is at the beginning of a word.

`\>`

This matches the empty string, provided it is at the end of a word.

`\w`

This matches any word-constituent character determined by the editor syntax table.

`\W`

This matches any character that is not a word-constituent.

`\scode`

This matches any character whose syntax is code (a character that represents syntax code)

`\Scode`

This matches any character whose syntax is not code.

© 2020 All Rights Reserved.

Reference

*GNU Emacs Manua*l by Richard Stallman (Cambridge: Free Software Federation, 1987)

© 2020 All Rights Reserved.

E

Abstract Generator Error and Warning Messages

This appendix contains additional information for error and warning messages issued by Abstract Generator. Messages that are fully explained in the message text itself are not included.

The purpose of this appendix is to provide further assistance in explaining and resolving error and warning messages that you might encounter when using Abstract Generator. This assistance includes supplementary explanations, workarounds, examples, and links to information elsewhere in the documentation.

For information about a particular message, search for the message ID in question: `ABS-XXX`.

- ABS-10500 to ABS-10599: Library Messages
- ABS-10000 to ABS-10099: Technology Messages
- ABS-11000 to ABS-11099: Cell Messages
- ABS-11800 to ABS-11899: Layout Messages
- ABS-12000 to ABS-12099: Pin Messages
- ABS-13000 to ABS-13099: Extract Messages
- ABS-14000 to ABS-14099: Abstract Messages
- ABS-16000 to ABS-16199: Import/Export Messages
- ABS-17000 to ABS-17099: Messages
- ABS-11900 to ABS-11999: System Messages
- ABS-11500 to ABS-11599: OAToolbox Messages
- ABS-20000 to ABS-20099
- ABS-23000 to ABS-23099

© 2020 All Rights Reserved.

ABS-10500 to ABS-10599: Library Messages

ABS-10517

Message: The *name* library has been interrupted while loading. *value* cells out of *value* have been successfully loaded.

Description: You have interrupted Abstract Generator while it was loading a cell library.

ABS-10526

Message: Failed to attach the *name* technology library to the *name* library. *technology warning*.

Troubleshooting: Check the following.

- Do not attempt to use the `absAttachTechLib` command without setting a library. You can do this only in `nogui` or `replay` modes.
- Ensure you have permission to write in the design library, specifically the `prop.xx` file, which contains property information regarding the attached technology library.
- Ensure that the design library exists and that it is not locked.
- Ensure that the technology library you are trying to attach exists.

© 2020 All Rights Reserved.

ABS-10000 to ABS-10099: Technology Messages

For additional information about technology error and warning messages in Abstract Generator, see the following:

- For general information about using a Virtuoso design environment technology file in Abstract Generator, see Appendix G, “Technology File Requirements for Abstract Generation”.
- For information about the technology file requirements of Abstract Generator, see Appendix F, “Tutorial: Generate Abstracts by Using the Standalone Abstract Generator”.
- For information about the Technology File Manager, technology files, and their construction, see *Virtuoso Technology Data ASCII Files Reference*.

ABS-10010

Message: No manufacturing grid value is specified in the technology file. Using the default `value` value. You can define the manufacturing grid value in the *physicalRules* section of the technology file.

Description: If you do not specify a default manufacturing grid value in the technology file, Abstract Generator uses a default value of 0.005.

Troubleshooting: You can set the manufacturing grid value in the `controls` section; for example,

```
controls(
  mfgGridResolution(
  ( 0.005000 )
  ) ;mfgGridResolution
) ;controls
```

ABS-10013

Message: The `name` layer is not defined in the techLayer class of the technology file.

Description: At least one of the required layers is not defined in the technology file.

Troubleshooting: Use the Technology File Manager to define the missing layers in the `techLayers` section of the technology file.

© 2020 All Rights Reserved.

ABS-10015

Message: There are insufficient metal layers defined in the current design. You must define at least two metal layers in the *validLayers* subsection of *LEFDefaultRouteSpec* constraint group of the technology file. These layers must have the layer function *metal* in the *functions* section. Update the technology file and attach it again.

Description: There must be at least two valid metal layers defined in the technology file.

Troubleshooting: Use the Technology File Manager to define these layers in the `validLayers` subsection of `LEFDefaultRouteSpec` constraint group of the technology file. In addition, ensure that these layers are assigned the function `metal` in the `functions` subsection of the `layerRules` section.

Use the Technology File Manager to ensure that the layer has been defined in all the sections listed below.

- `layerRules`

```
layerRules(
 functions(

  ;( layer     function   [maskNumber])
  ;( -----     --------   ------------)

   ( metal1    "metal"      1          )
   ( metal2    "metal"      2          )

 ) ;functions
) ;layerRules
```

- `LEFDefaultRouteSpec`

```
constraintGroups(
 ( "LEFDefaultRouteSpec"
   ; layer constraints
   interconnect(
     ( validLayers ( METAL1 METAL2 METAL3 )   )
   ) ;interconnect
 ) ;LEFDefaultRouteSpec
) ;constraintGroups
```


© 2020 All Rights Reserved.

ABS-10016

Message: Layer `name` has not been defined in all the required technology file classes. Ensure that the layer is defined in the *techLayers* and *techLayerPurposePriorities* sections of the technology file. Ensure also that it is listed in the *LEFDefaultRouteSpec* and that it has layer function metal in the *layerfunctions* section.

© 2020 All Rights Reserved.

Troubleshooting: Use the Technology File Manager to ensure that the layer has been defined in all the sections listed below.

- `layerRules`

```
layerRules(
 functions(

  ;( layer     function   [maskNumber])
  ;( -----     --------   ------------)
   ( metal1    "metal"      1          )
   ( metal2    "metal"      2          )

 );functions
);layerRules
```

- `LEFDefaultRouteSpec`

```
constraintGroups(
 ( "LEFDefaultRouteSpec"
   ; layer constraints
   interconnect(
     ( validLayers ( METAL1 METAL2 METAL3 )   )
   ) ;interconnect
 ) ;LEFDefaultRouteSpec
) ;constraintGroups
```

- `techLayers`

```
layerDefinitions(
 techLayers(
  ;( LayerName             Layer#     Abbreviation )
  ;( ---------             ------     ------------ )
  ;User-Defined Layers:
   ( metal1                23         metal1       )
   ( metal2                27         metal2       )
 ) ;techLayers
) ;layerDefinitions
```

- `techLayerPurposePriorities`

```
layerDefinitions(
 techLayerPurposePriorities(
  ;layers are ordered from lowest to highest
  ;priority
  ;( LayerName                 Purpose    )
  ;( ---------                 -------    )
   ( metal1                    drawing    )
   ( metal1                    pin        )
   ( metal1                    net        )
   ( metal1                    boundary   )
   ( metal1                    label      )
   ( metal2                    drawing    )
   ( metal2                    pin        )
   ( metal2                    net        )
   ( metal2                    boundary   )
   ( metal2                    label      )
 );techLayerPurposePriorities
);layerDefinitions
```


© 2020 All Rights Reserved.

Troubleshooting: Use the Technology File Manager to define all the required via layers in the `standardViaDefs` section. For any via layer, `layer1` is always the routing layer below and `layer2` is always the routing layer above; for example,

```
viaDefs(
 standardViaDefs(
 ;(viaDefName layer1 layer2 (cutLayer cutWidth
 ;cutHeight [resistancePerCut])
 ;(cutRows cutCol (cutSpace))
 ;(layer1Enc) (layer2Enc) (layer1Offset)
 ;(layer2Offset) (origOffset)
 ;[implant1 (implant1Enc) [implant2
 ;(implant2Enc)]])
 ;( ---------------------------------------------
 ------------------------------)

  ( VIA12  METAL1  METAL2  ("VIA1" 0.2 0.2)
   (1 1 (0.0 0.0))
   (0.1 0.1) (0.2 0.2) (0.0 0.0) (0.0 0.0)
   (0.0 0.0)
  )
 ) ;standardViaDefs
) ;viaDefs
```

Additionally, you must list the via layer in the `functions` subclass of the `layerRules` section and ensure that it has the function `cut`; for example,

```
layerRules
 functions(
  ;( layer   function   [maskNumber])
  ;( -----   --------   ------------)

   ( Via1    "cut"       5          )
 ) ;layerFunctions
) ;layerRules
```

ABS-10018

Message: The *`name`* via layer is not defined in all the required technology file sections. Ensure that the via layer is listed in the *functions* subsection of the *layerRules* section with the function set to *'cut'* and then try again.

Type: ERROR

© 2020 All Rights Reserved.

Troubleshooting: Use the Technology File Manager to define all the required via layers in the `standardViaDefs` section. For any via layer, `layer1` is always the routing layer below and `layer2` is always the routing layer above; for example,

```
viaDefs(
 standardViaDefs(
 ;(viaDefName layer1 layer2 (cutLayer cutWidth
 ;cutHeight [resistancePerCut])
 ;(cutRows cutCol (cutSpace))
 ;(layer1Enc) (layer2Enc) (layer1Offset)
 ;(layer2Offset) (origOffset)
 ;[implant1 (implant1Enc) [implant2
 ;(implant2Enc)]])
 ;( ---------------------------------------------
 ------------------------------)

  ( VIA12  METAL1  METAL2  ("VIA1" 0.2 0.2)
   (1 1 (0.0 0.0))
   (0.1 0.1) (0.2 0.2) (0.0 0.0) (0.0 0.0)
   (0.0 0.0)
  )
 ) ;standardViaDefs
) ;viaDefs
```

Additionally, you must list the via layer in the `functions` subclass of the `layerRules` section and ensure that it has the function `cut`; for example,

```
layerRules
 functions(
  ;( layer   function   [maskNumber])
  ;( -----   --------   ------------)

   ( Via1    "cut"       5          )
 ) ;layerFunctions
) ;layerRules
```

ABS-10019

Message: Failed to initialize the technology file because the *`name`* layer does not have the *`name`* purpose defined. Define the missing layer property in the *techLayerPurposePriorities* section of the technology file and reattach the technology file.

Description: If you want Abstract Generator to extract through a particular layer, the layer in question must have purposes `pin`, `drawing`, and `net` defined for it in the technology file.

© 2020 All Rights Reserved.

Troubleshooting: Use the Technology File Manager to define the missing layer property in the `techLayerPurposePriorities` section of the technology file; for example,

```
techLayerPurposePriorities(
 ;layers are ordered from lowest to highest priority
 ;( LayerName                 Purpose    )
 ;( ---------                 -------    )

  ( metal1                    drawing    )
  ( metal1                    pin        )
  ( metal1                    net        )
  ( metal1                    boundary   )
 ) ;techLayerPurposePriorities
```

ABS-10021

Message: Though the *`name`* layer is defined in the *spacings* section of the *foundry* constraint group, its definition is missing from one or more of these sections in the technology file: *techLayers, techLayerPurposePriorities, techDisplays; functions* subsection in the *layerRules* section; or *interconnect* section of the *LefDefaultRouteSpec* constraint group. Update the technology file and try again.

Description: A layer needs to be defined in all the following sections of the technology file:

- `techLayers`, `techLayerPurposePriorities`, and `techDisplays`
- `functions` subclass in the `layerRules` section
- `interconnect` constraint category in the `LefDefaultRouteSpec` constraint group
- `spacings` constraint category in the `foundry` constraint group section

A layer is regarded as not fully defined if it is not defined in any of these sections.

For other related error message, see ABS-10016.

Troubleshooting: Use the Technology File Manager to define the layer in all the required sections in the technology file. Save and reload the file.

© 2020 All Rights Reserved.

ABS-10022

Message: The `width`, `pitch`, `separation`, and `offset` values of all metal layers must be a multiple of the current manufacturing grid value. Layer *`name spacing`* value is not a multiple of the current manufacturing grid value. Update the spacing value for this layer in the *spacing* section of the *foundry* constraint group and the *routingGrid* section of the *LefDefaultRouteSpec* constraint group in the technology file.

Description: All metal layer width, pitch, separation, and offset values must be a multiple of the current manufacturing grid value.

© 2020 All Rights Reserved.

Troubleshooting: Use the Technology File Manager to correct the values.

- The manufacturing grid is set in the controls section. The width and spacing values are set in the `spacings` constraint category of the `foundry` constraint group; for example,

```
controls(
  mfgGridResolution(
  ( 0.005000 )
  ) ;mfgGridResolution
) ;controls

constraintGroups(
 ( "foundry"
   ;physical constraints
   spacings(

   ;( constraint layer1 layer2 value )
   ;( ---------- ------ ------ ----- )

    ( minWidth   "metal1"      0.5 )
    ( minSpacing "metal1"      0.5 )
    ( minWidth   "metal2"      0.5 )
    ( minSpacing "metal2"      0.5 )
   ) ;spacings
 ) ;foundry
) ;constraintGroups
```

- The routing pitch and routing offset values are set in the `routingGrids` constraint category `LEFDefaultRouteSpec` constraint group; for example,

```
constraintGroups(
 ( "LEFDefaultRouteSpec"
   ; physical constraints
   routingGrids(
     ( rightDiagPitch   METAL5    0.01 )
     ( rightDiagOffset  METAL5    0.05 )
     ( leftDiagPitch    METAL5    0.2  )
     ( leftDiagOffset   METAL5    0.4  )
   ) ;routingGrids
 ) ;LEFDefaultRouteSpec
) ;constraintGroups
```

ABS-10023

Message: The *`name`* layer does not have a minimum width. Define in the *spacings* section of *foundry* constraint group of the technology file and then try again.

© 2020 All Rights Reserved.

Troubleshooting: Use the Technology File Manager to define the minimum width for a layer in the `spacings` section of `foundry` constraint group; for example,

```
constraintGroups(
 ( "foundry"
   ;physical constraints
   spacings(

   ;( constraint layer1 layer2 value )
   ;( ---------- ------ ------ ----- )

    ( minWidth   "metal1"      0.5 )
   ) ;spacings
 ) ;foundry
) ;constraintGroups
```

ABS-10024

Message: The *`name`* layer does not have a minimum spacing value defined. The entries for `minSpacing` table are either non-applicable or the default value is not given. Define appropriately in the *spacings* section of *foundry* constraint group of the technology file and then try again.

Description: Abstract Generator stops because the `minSpacing` value set for the given layer is 0.

Troubleshooting: Use the Technology File Manager to set a valid `minSpacing` value for the layer; for example,

```
constraintGroups(
 ( "foundry"
   ;physical constraints
   spacings(

   ;( constraint layer1 layer2 value )
   ;( ---------- ------ ------ ----- )

    ( minWidth   "metal1"      0.5 )
   ) ;spacings
 ) ;foundry
) ;constraintGroups
```

Abstract Generator uses this value for the minimum spacing for both metal and via layers.

ABS-10028

© 2020 All Rights Reserved.

Message: The routing-grid offset value of the `name` layer is not equal to the routing-grid offset value for base layer `name`.

Description: The routing offset value for a layer must be equal to that of the equivalent base layer; for example, `metal2` or `metal1`, depending on which layer is being examined.

Troubleshooting: Use the Technology File Manager to define the offset value in the `LEFDefaultRouteSpec` constraint group of the technology file.

ABS-10029

Message: No via is defined for the `name` layer in the technology file, which is preventing extraction through this layer. Define the via in the following sections in the technology file: *viaDefs (standardViaDefs or customViaDefs)* and *validVias* in the *interconnect* section of the *LefDefaultRouteSpec* constraint group.

Type: WARNING

Description: If you define a layer above a routing layer, you must define a via layer to go between them. Abstract Generator cannot extract connectivity without this via layer.

Troubleshooting: Use the Technology File Manager to define the via layer in all the required technology file sections.

ABS-10030

Message: No default via is defined for the `name` layer. Ensure that the technology file includes a fully defined via layer for the layer.

© 2020 All Rights Reserved.

Troubleshooting: To correctly define a default via in the OpenAccess technology file,

- Define the via in the `customViaDefs` section of the technology file; for example,

```
viaDefs(
 customViaDefs(
  ;(viaDefName libName cellName viewName
  ; layer1     layer2  resistancePerCut)

   ( M2_M1      lib     M2_M1    via
     metal1     metal2  4.0 )
 ) ;customViaDefs
) ;viaDefs
```

  The cellview used in the definition must exist in the technology library. Symbolic and layout cellviews usually correspond to a device defined in the `devices` section. If the cellview does not exist, you can generate one by adding a device to the `devices` section. If you want to use a via cellview, you need to create it manually or use `lef2oa`, which will create it automatically.

- Ensure that the via is listed in the `LEFDefaultRouteSpec`; for example,

```
constraintGroups(
 ( "LEFDefaultRouteSpec"
   ; layer constraints
   interconnect(
     ( validVias ( M1_M2 M2_M1 )   )
   ) ;interconnect
 ) ;LEFDefaultRouteSpec
) ;constraintGroups
```

- List the via layer in the `functions` subclass of the `layerRules` section and ensure that it has function `cut`; for example,

```
layerRules
 functions(
  ;( layer   function   [maskNumber])
  ;( -----   --------   ------------)

   ( v1      "cut"       5          )
 ) ;layerFunctions
) ;layerRules
```

ABS-10034

© 2020 All Rights Reserved.

Message: There are errors in the technology file that need to be resolved before proceeding to the abstract generation steps.

Description: There are a number critical checks performed by Abstract Generator on start-up, among them a validity check on the technology data associated with a design. If any of these checks fails, Abstract Generator issues an appropriate error message.

Troubleshooting: Use the Technology File Manager to resolve these errors before proceeding.

ABS-10042

Message: Technology initialization failed because the *name* layer with the *value* layer number and the *name* layer with the *value* layer number have the same name. The same layer name cannot be associated with two layer numbers. Rectify the error in the technology file and then try again.

Description: Each layer has an associated layer name and is assigned a unique layer number in `techLayers` in the `layerDefinitions` section of the technology file. The same layer name cannot be associated with two layer numbers.

ABS-10044

Message: Failed to perform `antennaSideArea` calculations because the *name* layer does not have a thickness value specified. Specify the thickness value in the *antenna* subsection of the *foundry* constraint group of the technology file.

Troubleshooting: To perform antenna side area factor or ratio calculations, set a `thickness` value for the layer in the `antenna` subsection of `foundry` constraint group.

ABS-10046

© 2020 All Rights Reserved.

Message: Failed to initialize Abstract Generator because the layer *name* in via layer *layer1Name*-*cutLayerName*-*layer2Name* must be a routing or masterslice layer. Define the layer in the appropriate section, either *validLayers* section of *LefDefaultRouteSpec* constraint group or *functions* section of *layerRules* section.

Troubleshooting: Use the Technology File Manager to update your technology file accordingly.

You must define masterslice layers in the `functions` subsection of the `layerRules` section. Specify the `function` as one of `ndiff`, `nplus`, `nwell`, `pdiff`, `poly`, `pplus`, `pwell`, or `li`. For more information, see the *Virtuoso Technology Data ASCII Files Reference*.

ABS-10047

Message: Failed to initialize Abstract Generator because either the layer2 slot in via layer *layer1Name*-*cutLayerName*-*layer2Name*, which is currently *layer2Name*, must be a routing layer, else the layers might appear in the wrong order in LEF. Define the layer in the *validLayers* section of the *LefDefaultRouteSpec* constraint group in the technology file.

Troubleshooting: Use the Technology File Manager to define the layer correctly in the `validLayers` section of `LEFDefaultRouteSpec` constraint group. The layers must be specified from lowest to highest, otherwise the technology checker might indicate that you have the layers in the wrong order.

Note: If the layers are in the wrong order, you need to define the vias in the bottom-up order; first layer bottom, second layer cut, and third layer top of the via. The lower level layer must be listed first in `standardViaDefs`; for example, `Poly-Cont-Metal1`.

ABS-10048

Message: Failed to initialize Abstract Generator because either the *name* layer for the *name* via is not correctly defined. Ensure that the layer is correctly specified in the *standardViaDefs* subsection of the *ViaDefs* section and in the *techLayers* subsection of the *layerDefinitions* section in the technology file and then try again.

© 2020 All Rights Reserved.

Type: ERROR

Troubleshooting: **Note:** Information about via definitions is included in `standardViaDefs` in the `viaDefs` section.

There is an error in the via layer definition for the specified via. The layer in question is either missing from `standardViaDefs` section or is not defined in the `techLayers` section.

1. Ensure that the layer in question is listed correctly in the `standardViaDefs` section; for example,

```
viaDefs(
 standardViaDefs(
 ;(viaDefName layer1 layer2 (cutLayer cutWidth
 ;cutHeight [resistancePerCut])
 ;(cutRows cutCol (cutSpace))
 ;(layer1Enc) (layer2Enc) (layer1Offset)
 ;(layer2Offset) (origOffset)
 ;[implant1 (implant1Enc) [implant2
 ;(implant2Enc)]])
 ;( ---------------------------------------------
 ------------------------------)

  ( VIA12  METAL1  METAL2  ("VIA1" 0.2 0.2)
   (1 1 (0.0 0.0))
   (0.1 0.1) (0.2 0.2) (0.0 0.0) (0.0 0.0)
   (0.0 0.0)
  )
 ) ;standardViaDefs
) ;viaDefs
```

   Note: For any via layer, `layer1` is always the lower routing layer and `layer2` is always the upper routing layer

2. Ensure that all the layers listed are defined in the `techLayers` section; for example,

```
layerDefinitions(
 techLayers(
  ;( LayerName              Layer#     Abbreviation )
  ;( ---------              ------     ------------ )
  ;User-Defined Layers:
   ( metal1                 23         metal1       )
   ( v1                     25         v1           )
   ( metal2                 27         metal1      )
   ( v2                      32         v1          )
   ( metal3                 34         metal1       )
   ( v3                     35          v1
)   ( metal4                 36         metal1
)
   ( v4                     52          v1
)   ( metal5                 53         metal1
)
 ) ;techLayers
) ;layerDefinitions
```


© 2020 All Rights Reserved.

ABS-10050

Message: Failed to create the *name*, *purpose* layer-purpose pair in the technology file. Ensure that you have permissions to write to the technology file and then try again.

Description: The layer-purpose pair information is required during the abstract generation process, and it is specified in the `techPurposes` construct in the `layerDefinitions` section. If missing, you can append this information to the technology file. If write permissions exist on the technology file, Abstract Generator will automatically add the missing layer-purpose pair. However, you would not be able to update the technology file if you do not have the write permission on the file, which is when Abstract Generator issues this message.

Troubleshooting: Change the permission setting of the technology file or contact the person responsible for updating the technology file. Use the Technology File Manager to add the missing layer-purpose definition to the technology file prior to invoking Abstract Generator.

ABS-10053

Message: Failed to set the *name* property because *reason*.

Description: Abstract Generator has failed to set the required property.

Troubleshooting: If you are unable to isolate the problem, contact your Cadence Customer Support representative for help.

ABS-10054

Message: Failed to initialize Abstract Generator because either the *name* layer is not defined in the *standardViaDefs* subsection of the *viaDefs* section or the layer function has not been identified as *'cut'* in the *functions* subsection of the *layerRules* section. Ensure that the layer is correctly defined in both *standardViaDefs* and *functions* subsection and then try again.

Type: ERROR

© 2020 All Rights Reserved.

Description: You need to define via layers in the `standardViaDefs` section and as ‘cut’ in the `functions` subclass of the `layerRules` section. This error will be generated if the via layer is not defined in any one of these.

Troubleshooting: Using the Technology File Manager, check the `standardViaDefs` section and the `functions` subsection in the `layerRules` section in the technology file and update the section in which the via layer definition is missing.

ABS-10055

Message: Failed to create the *`name`* site in the *`technology library name`* technology file.

Description: Abstract Generator tried unsuccessfully to create a site in the OpenAccess technology file, `tech.db`. The exception message issued by OpenAccess indicates the reason for the failure. For example, you might not have permission to update the `tech.db` file.

ABS-10057

Message: Release IC 6.1.0 onwards, Abstract Generator will not support the "USEMINSPACING OBS ON" property in the technology file. This property will be ignored.

Type: WARNING

ABS-10058

Message: Release IC 6.1.0 onwards, Abstract Generator will not support the "USEMINSPACING PIN ON" property in the technology file. This property will be ignored.

Type: WARNING

ABS-10059

© 2020 All Rights Reserved.

Message: The layer `layerName`, defined in the `spacingTables` section of the *foundry* constraint group, is also defined in the *techDerivedLayers* section of the technology file. The abstract generator does not support derived layers and this layer will, therefore, be ignored. As a result, the abstracts generated will not have any shape on this layer.

Troubleshooting: For more information, see Layer with Spacing Rules is a Derived Layer.

ABS-19006

Message: The routing direction for layer `layerName` has not been specified. Specify the appropriate direction in the *routingDirections* sub-section of *layerRules* section of the technology file.

Type: Error

© 2020 All Rights Reserved.

ABS-11000 to ABS-11099: Cell Messages

ABS-11001

Message: Cell `name`: The `view name` view has been removed.

Description: Cellviews are created upon the successful completion of each flow step in the abstract generation process. This message is generated when a cell view has been removed outside of Abstract Generator.

Troubleshooting: You can recreate the missing views by importing either GDSII (*File – Import – Stream (GDSII)*) or DEF data (*File – Import – DEF*). Alternatively, you can run the appropriate abstract generation steps.

ABS-11004

Message: Cell `name`: Failed to open the `view name` view because `message`.

Description: This error is issued when you try to open a non valid cell view.

Troubleshooting: Check the cell and remove from it the library if required.

ABS-11011

Message: Cell `name:` The direction of the `name` terminal has changed from `direction` to `direction`. This change will invalidate any abstract generation steps already run. The terminal directions can be modified in the *Terminal Properties* form.

Description: The specified terminal has had its direction changed since the previous steps were run for this cell.

Troubleshooting: For information about setting terminal directions, see Viewing and Editing Cell Terminal Properties.

ABS-11012

© 2020 All Rights Reserved.

Message: Cell *name:* The type of the *name* terminal has changed from *termType* to *termType*. This change will invalidate any abstract generation steps already run. The terminal type can be modified in the *Terminal Properties* form.

Description: The specified terminal has had its type changed since the previous steps were run for this cell.

Troubleshooting: If this change was not intentional, use the Terminal Properties form to restore the terminal type to its original setting. For information about setting terminal types, see Viewing and Editing Cell Terminal Properties.

ABS-11013

Message: Cell *name*: The shape of the *name* terminal has changed from *shape* to *shape*. This change will invalidate any abstract generation steps already run. The terminal shape can be modified in the *Terminal Properties* form.

Description: The terminal shape has been changed since the previous steps were run for this cell.

Troubleshooting: If this change was not intentional, use the Terminal Properties form to restore the terminal type to its original setting. For information about setting terminal shapes, see Viewing and Editing Cell Terminal Properties.

ABS-11024

Message: Failed to process the cell because the *step name* view *reason*. Check the cell directory and close all applications that is using the same step and then try again.

Description: This error message indicates that Abstract Generator cannot process the cell because it might be locked by another application. So the error message states the abstract generation flow step name from Pins, Extract, or Abstract that might be locked by another application.

Troubleshooting: Check the cell directory and remove any locked files or close an application that is using the same flow step.

© 2020 All Rights Reserved.

ABS-11800 to ABS-11899: Layout Messages

ABS-11804

Message: You have chosen to perform either signal, power, or antenna extraction, but there are shapes on purpose boundary in your layout view. Large boundary shapes can cause extraction problems, such as shorts, but can be remedied by excluding the purpose boundary by using *General options* dialog box or by entering the following (example) geometry specification: `metal1 -> (metal1 drawing) or (metal1 pin) or (metal1 net)` in the appropriate GSpec location in the Extract step.

Description: You have selected one of the extraction options (i.e., signal, power, or antenna extraction) and there are shapes on purpose `boundary` in the layout view for at least one of the layers entered in an extract geometry specification (GSpec) table.

For some designs, you will not need to take any action. However, if your design contains large shapes on purpose `boundary`, this can cause extraction problems (such as shorts in a design) which are only discovered after a protracted extraction process.

Troubleshooting: Remove purpose `boundary` from the geometry specification. For example, instead of specifying `metal1`, type the following instead.

```
(metal1 drawing) or (metal1 pin) or (metal1 net)
```

Even if you do not perform extraction, the pins created in the Abstract step could still be affected if you have blockage shapes on the layout.

© 2020 All Rights Reserved.

ABS-12000 to ABS-12099: Pin Messages

ABS-12002

Message: Cell `name`: No terminals have been created. Use the *Map text labels to pins* option in the *Map* tab of the *Pins* step to define cell pins.

Troubleshooting: For information about generating pins, see Mapping Text Labels to Pins.

ABS-12004

Message: Cell `name`: The abstract generator could not generate pins because the *Map text labels to pins* field did not contain valid values. You can proceed with the Extract step. However, to generate pins in the abstract view, specify valid values in the *Map text labels to pins* field and rerun the Pins step.

Troubleshooting: For information about generating pins, see Mapping Text Labels to Pins.

ABS-12006

Message: Cell `name`: No pin will be created for the text label '`label`' at (`x`,`y`) because no overlapping geometry exists for this text label. The abstract generator creates pins corresponding to geometries that overlap the text labels specified in the *Map text labels to pins* field on the *Map* tab. Therefore, while updating this field with pin mapping specification, ensure that geometries exist below the text labels specified. You can also check to ensure that the GDSII layout data has been imported into the abstract generator on correct layers.

Description: Abstract Generator creates pins corresponding to geometries that overlap the text labels specified in the *Map text labels to pins* field of the Pins step. This error message is generated when Abstract Generator does not find any shape overlapping the text label specified in the *Map text labels to pins* field.

© 2020 All Rights Reserved.

ABS-12007

Message: Cell `name`: A pin has already been created on the net `name`, which conflicts with the label `name` at (`x`,`y`). Ensure that the *Map text labels to pins* field on the *Map* tab has been set correctly. You can also check to ensure that the GDSII layout data has been imported into the abstract generator on correct layers.

Description: A pin has already been created with the same name on the same layer.

ABS-12008

Message: Cell `name`: No value has been specified in the *Map text labels to pins* field on the *Map* tab for the bin `name`. Therefore, the default value `defaultvalue` will be used for mapping text labels to pins.

Description: Unless there are already terminals present, the *Map text labels to pins* specification is required to generate a valid pins view. If you do not specify this, Abstract Generator automatically determines a suitable default value before processing.

ABS-12009

Message: The *Map text labels to pins* field has been left blank and the abstract generator is unable to determine a default text pin mapping for the bin `name` because the cell layout does not contain text labels intersecting any geometry. The abstract generator tries to calculate a default value by examining the layers with text labels and then finding the metal layers that have shapes that intersect any of the text labels. If the abstract generator finds a shape under a text label on a metal layer, it sets the *Map text labels to pins* option to map those text labels to that metal layer. In the current situation, the abstract generator cannot find any metal shapes under the text labels.

ABS-12010

© 2020 All Rights Reserved.

Message: Cell `name`: The prBoundary cannot be calculated even though the prBoundary creation has been enabled on the *Boundary* tab. This is because no layers have been specified in the *Using geometry on layers* field on the *Boundary* tab. Specify the appropriate layers for prBoundary creation.

Troubleshooting: For more information, see Specifying the Layers for PR Boundary Calculation.

ABS-12011

Message: Cell `name`: No prBoundary has been created even though the prBoundary creation has been enabled on the *Boundary* tab. This is because no geometry exists on the layers specified in the *Using geometry on layers* field on the *Boundary* tab. Specify the appropriate layers for prBoundary creation.

Troubleshooting: For information about specifying layers to be used for boundary calculations, see Specifying the Layers for PR Boundary Calculation.

ABS-12012

Message: Cell `name`: Multiple prBoundaries have been created based on the value specified in the *Using geometry on layers* field on the *Boundary* tab. Update this value so that a single prBoundary is created.

Troubleshooting: Correct the value in the *Using geometry on layers* field to enable Abstract Generator to create a single PR boundary.

ABS-12013

Message: Cell `name`: the calculation of the prBoundary failed even though the prBoundary creation has been enabled on the *Boundary* tab. Ensure that the layers on which the calculation is to be based are specified in the *Using geometry on layers* field on the *Boundary* tab. The specified layers should lead to the creation of a single prBoundary. Additionally, ensure that geometry exists on the specified layers.

© 2020 All Rights Reserved.

Troubleshooting: For information about specifying layers to be used for boundary calculations, see Specifying the Layers for PR Boundary Calculation.

See also ABS-12010 and ABS-12011.

ABS-12015

Message: Cell `name`: the prBoundary does not enclose all cellview geometries. This is because not all layers with geometry for calculating prBoundary have been specified in the *Using geometry on layers* field on the *Boundary* tab. Ensure that all the required layers have been specified in this field.

Troubleshooting: For more information, see Specifying the Layers for PR Boundary Calculation.

ABS-12016

Message: The *Create boundary* option on the *Boundary* tab has been set to off, but no prBoundary exists in cell. Set the *Create boundary* option to either 'always' or 'as needed'.

Description: The *Create boundary* option should be set to *off* only if a prBoundary already exists in a cell. This message is displayed if no boundary exists and there is no request to create one.

Troubleshooting: For information about creating place-and-route boundaries and specifying the layers to be used in the necessary calculations, see Boundary Tab.

ABS-12019

Message: Cell `name`: database could not create pin named `pin name`. Verify the permissions on the design and ensure that write permission exists.

Description: The database is unable to create a pin with the specified name. This is because the write permissions on the design does not exist.

© 2020 All Rights Reserved.

ABS-12020

Message: Cell `name`: database could not create net named `net name`. Verify the permissions on the design and ensure that write permission exists.

Description: The database is unable to create a net with the specified name. This is because the write permissions on the design does not exist.

ABS-12022

Message: Cell `name`: the existing terminal `terminalname` was ignored based on the *Exclude existing terminals* field on the *Map* tab. However, no equivalent terminal was created based on the option specified in the *Map text labels to pins* field.

Description: This warning is issued when the terminals specified in the *Exclude existing terminals* field on the *Map* tab are not created back from the equivalent text labels existing in the layout.

ABS-12023

Message: Cell `name`: the pin *name* has shape on an invalid user layer numbered *number* in the technology file. The pin will be deleted from the layout.

Description: This warning is issued during the Pins step when Abstract Generator comes across a pin that exists on an invalid technology layer. Abstract Generator deletes such pins from the layout after issuing this warning.

ABS-12024

Message: Cell `name`: no pin shape will be created on the layer *name* because the layer does not have pin purpose defined in the technology file.

© 2020 All Rights Reserved.

Description: This error message is issued when Abstract Generator attempts to add pin shapes on a layer that does not have a purpose defined in the technology file.

Abstract Generator might attempt to create pins in the following ways:

- From the text labels that overlap shapes on the specified layer-purpose pairs. For more information, see Mapping Text Labels to Pins.
- From preexisting terminals and pins.
- From the figures and instance pins attached to a terminal's net. This happens when a cellview contains terminals but no pins.

Troubleshooting:

- If you want the shapes to be added on the specified layer, define the missing LPPs in the `techLayerPurposePriorities` section of the technology file by using the Technology File Manager. Also see ABS-10050.
- If you want Abstract Generator to ignore the existing terminals without pins, do one of the following:
  - Remove the nets associated to the terminals from the cellview.
  - Specify the terminal names in the *Exclude existing terminals* field on the Map tab.
- Check the value specified for the *Map text labels to pins* field if the error occurs when the pins are being generated through text labels.

ABS-12025

Message: Cell *`name`*: Terminals *`name`* and *`name`* are must join terminals but have been mapped to different terminal types. *`terminalname`* has been specified as *`type`* terminal and *`terminalname`* has been specified as *`type`* terminal. Check the regular expressions specified on the *Map* tab.

© 2020 All Rights Reserved.

ABS-12027

Message: Cell `name`: the existing terminal *name* specified in the *Exclude existing pins on layers* field does not have any pin shape. This terminal will be deleted. The equivalent terminal will be created based on the value specified in the *Map text labels to pins* field on the *Map* tab.

Description: This warning message is issued when a terminal on the layer specified in the *Exclude existing pins on layers* field does not have a pin.

© 2020 All Rights Reserved.

ABS-13000 to ABS-13099: Extract Messages

ABS-13006

Message: Cell `name`: layer 'POLY' is not listed in the *Layer Assignment for* `name` *Extraction* table. This layer is required for antenna extraction. Select 'POLY' in the *Layer* column and specify other information for this layer to be extracted.

Troubleshooting: For information about adding layers, see Assigning Layers for Signal Extraction and Table on page 155.

ABS-13007

Message: Cell `name`: layer 'DIFF', which is required for antenna extraction, is not listed in the *Layer Assignment for* `name` *Extraction* table. For antenna extraction, select 'DIFF' in the *Layer* column and specify the layer 'DIFF' as type '(DIFF andnot POLY)', where DIFF and POLY are the diffusion and polysilicon layers in question.

Troubleshooting: For information about adding layers, see Assigning Layers for Signal Extraction and Assigning Layers for Antenna Extraction.

ABS-13008

Message: Cell `name`: either the layer 'DIFF' is not specified as type '(diff diff andnot poly)' in the *Layer Assignment for* `name` *Extraction* table or the geometry specification for a layer contains two or more layers from the *Layer* column. Select 'DIFF' in the *Layer* column and specify it as type '(diff diff andnot poly)', where diff and poly are to be replaced with the real names of the diffusion and polysilicon layers in question. If a geometry specification uses a temporary layer to limit extraction and the temporary layer is a combination of two or more layers from the *Layer* column, do not select the individual layers in the *Layer* column.

Description: This message appears when a diffusion layer is not specified in one of the layer assignment for extraction tables. This message might also appear when a geometry specification contains two or more layers from the Layer column.

© 2020 All Rights Reserved.

Troubleshooting: For information about adding layers, see Assigning Layers for Signal Extraction and Assigning Layers for Antenna Extraction.

ABS-13009

Message: Extraction has been aborted because the number of shapes that were extracted for net *name* exceeds the current extraction limit of *value*. Use the `ExtractShapeLimit` option to increase the extraction limit and rerun the command.

Description: This situation might result because of one of the following reasons:

- The number of shapes being extracted from the current design exceeds the maximum permitted value. The default value is `10000`.
- The net has the wrong terminal type set.
- The design contains a short between nets.

Troubleshooting: You can take one of the following actions to overcome this error:

- Increase the extract shape limit by using the following setting either in the GUI or in the .abstract.options file:

  ```
  absSetOption( "ExtractShapeLimit" "<any positive number>" )
  ```

  For example,

  ```
  absSetOption( "ExtractShapeLimit" "400500" )
  ```

  Additionally, you can set the stack size for the abstract process to unlimited by typing the following in the terminal window before invoking Abstract Generator:

  ```
  limit stacksize unlimited
  ```

- Select *Cells – Terminal Properties* to open the *Terminal Properties* form and check that the terminal type is set correctly for the net in question.

ABS-13010

© 2020 All Rights Reserved.

Message: Cell `name`: extraction has been aborted because a short has been detected between the nets `name` and `name`. Check the following: 1) If you are performing antenna extraction, ensure that the *Layer Assignment for Antenna Extraction* table on the *Antenna* tab has been populated correctly. The geometry specification for the diffusion layer in the table should read 'diff diff and not poly'. 2) Ensure that the *Layer connectivity* on the *General* tab is set up correctly for all layers, especially for polysilicon and metal1 and diffusion and metal1, such as '(poly metal1 contact) (diff metal1 contact)'.

ABS-13012

Message: Cell `name`: a possible must join terminal *name* has been ignored because the terminal was not listed in the *Create Must Connect Pins* option on the *name* tab. Ensure that all must connects are joined for a specific terminal, either by activating the *Always* option on the *Signal* tab or listing the terminals in the *Only on terminals named* field.

ABS-13014

Message: Cell `name`: there is no pin geometry on the net `name`. To create pins on a specific layer, select the *Create Pins* check box corresponding to the layer in the *Layer Assignment for* `name` *Extraction* table.

Description: It is usually unnecessary to create pins on polysilicon, contact, or diffusion layers.

Troubleshooting: For information about creating pins, see Create Pins.

ABS-13018

Message: Cell `name`: net `name` process antenna calculations for terminal `name` on instance `name` have not been used.

© 2020 All Rights Reserved.

Description: Abstract Generator has failed to identify any extracted nodes as representing instance terminal pin shapes. Because the nodes have not been matched to an instance pin, the associated antenna calculations have not been taken into account. This problem could be a result of the net not being fully routed in DEF.

ABS-13019

Message: Cell `name`: net `name` process antenna `type` value has been ignored because it is different from other `layerName` values for terminal `name` on instance `name`. This might indicate that the antenna information stored on a terminal is inconsistent with itself. This could occur if there is an error in the antenna calculation or the property values have changed in the abstract. The inconsistency might also indicate that the pin area for the instance terminal has changed since it was first subtracted in the Adjust step.

ABS-13020

Message: Cell `name`: matching of the gate region with correct antenna oxide type has failed. This type of failure occurs when the abstract generator is unable to identify the diffusion layer for gate calculation. Check the settings in the *Layer Assignment for Antenna Regions* table on the *Antenna* tab.

© 2020 All Rights Reserved.

Description:

Abstract Generator is unable to identify which diffusion type a gate is over. This can occur if you specify the antenna regions like the one below.

```
Layer     Geometry Specification     Region     Oxide
-----     ----------------------     ------     -----
poly      poly and (diff or diff2)   Gate
diff      diff andnot poly           Drain      Oxide1
```

Where *poly*, *diff*, *diff2* and *Oxide1* are replaced by the real names of the layers in question.

Note: Any *Oxide* option can be set here (or even none), but the antenna style must be LEF 5.5.

In the above example *diff2* has not been specified as a drain region but has been specified in the *Geometry Specification*. This means that any gates over *diff2* will not be identified with a particular oxide.

Troubleshooting:

The fix for this example is to add the following extra line:

```
Layer     Geometry Specification     Region     Oxide
-----     ----------------------     ------     -----
poly      poly and (diff or diff2)   Gate
diff      diff andnot poly           Drain      Oxide1
diff2     diff2 andnot poly1         Drain      Oxide2
```

Another scenario that can cause this message to appear, is where there is a complicated size operator specified in the *Geometry Specification* for the polysilicon layer. If you have then sized up the diffusion layer in the *Gate* specification, it is possible that this could create a gate area which would not overlap the unsized diffusion layer.

For more information, see antenna Oxide.

ABS-13021

Message:

Cell *name*: a short has been detected between the nets *netName* and *netName* during antenna calculation. This indicates a problem in the design unless the nets are connected by a polysilicon resistor. Though the short, if it exists between the nets, is ignored for antenna calculation but is considered for the signal net extraction because either the *Use different layer assignments for antenna calculations only* option is selected or the extraction depth has been limited.

© 2020 All Rights Reserved.

Description: This is implemented in particular for antenna calculations for analog blocks where a polysilicon resistor is used to connect two nets. The short needs to be respected for the signal net extraction, but the antenna extraction needs to cross the resistor to find the gate area when performing cumulative antenna calculations.

Troubleshooting: To avoid these messages when performing antenna extraction in analog blocks, specify `polysilicon andnot resistor` in the *Layer Assignment for Signal Extraction* and *Layer Assignment for Power Extraction* tables. Remove the resistor part in the *Layer Assignment for Antenna Extraction* table.

ABS-13022

Message: Cell *`name`*: no pin shape will be created on the layer *name* because the layer does not have net purpose defined in the technology data. Ensure that the technology file is correct. Otherwise, check the design.

Description: This error message is issued when you try to add shapes on a layer that does not have a purpose defined in the technology file.

Troubleshooting: Use the Technology File Manager to define the missing LPPs in the `techLayerPurposePriorities` section of the technology file. Also see ABS-10050.

ABS-13031

Message: Cell *`name`*: Terminal *`name`* has been ignored although it has the mustJoin attribute. This is because the terminal either has the property LEF58_MUSTCONNECTALLPORTS or was listed with the Create MustConnectAllPins option or the Create mustConnectAllPinsTerminal option.

© 2020 All Rights Reserved.

ABS-14000 to ABS-14099: Abstract Messages

ABS-14003

Message: Cell *name*: the power ring for net *name* only contains *value* shapes that are within *value* microns of the boundary. There must be at least 4 shapes within this distance.

Description: To create a power ring, there must be at least four shapes within a specified distance of the cell boundary. This is determined using the shape closest to the cell boundary.

Troubleshooting: For more information about creating power rings, see Creating Ring Pins and Specifying the Distance of the Ring Pin from the Boundary.

ABS-14006

Message: There are no power and ground rails common to all cells in the current bin. Use the *Power Rail Widths and Offsets* option to specify the characteristics of the power and ground rails for the bin.

Description: During the Abstract step, the power rail analysis check could not identify the power or ground rails.

Troubleshooting: Use the *Power rail widths, offsets and shape* option to specify the power and ground rails. You might also need to redistribute cells into separate bins to complete this step.

ABS-14007

Message: Cell *name*: contains non-orthogonal geometry which cannot be modeled. There are *value* cells with non-orthogonal geometry.

Description: The design contains one or more cells containing diagonal geometry which Abstract Generator cannot model.

Abstract Generator usually adopts the stair-step coverage method in order to model 45-degree geometry, but this method failed on the named cells. It could be that the intersection of the 45-degree shape and the stair-step is too small to be processed by Abstract Generator.

© 2020 All Rights Reserved.

Troubleshooting: Increase the size of your stair step and run Abstract Generator again. If the problem persists, consider editing your layout data.

- For information about the stair-step model, see Setting Up Options for 45-Degree Geometry.
- For information about designing library data for use in Cadence flows, see Appendix C, "Library Design Guidelines."

ABS-14008

Message: Cell `name`: terminal `name` shape `name` is different from the calculated value `value`. You can reset shape properties using the *Terminal Properties* form.

Description: The shape value calculated for the named terminal is different from the value already stored in the cellview. Abstract Generator is unable to reset the property value.

Troubleshooting: Reset the value manually using the *Terminal Properties* form.

ABS-14013

Message: Cell `name`: The blockage type selected for the layer `name` in the *Layer Assignment for the Blockages* table is *Shrink*. However, no value is specified for either *Shrink Track* or *Shrink Dist*. If you do not specify a value in either of these columns corresponding to the layer, the abstract generator will use the default value of 2 tracks for creating the blockage. This default value is small and will increase the run time of the Abstract step significantly. If you want to avoid this, you must specify either the *Shrink Track* or *Shrink Dist* value or both.

© 2020 All Rights Reserved.

Description: If you choose the type of blockage you want create on a layer as `Shrink` in the *Blockage* column of the *Layer Assignment for Blockages* table on the *Blockage* tab (Abstract step), you need to specify a distance value (in microns) in the *Shrink Dist* field. Alternatively, you can specifiy a value in terms of the number of tracks between the blockages in the *Shrink Track* field. Abstract Generator combines blockages that are within the specified distance from each other or are separated by the specified number of tracks by filling the space between them with more blockage.

If you specify both `Shrink Track` and `Shrink Dist`, the values are added and blockages separated by a distance less than or equal to the total distance are merged. If you do not supply a distance, Abstract Generator uses a default value of `2` tracks and displays this warning message.

ABS-14015

Message: Cell *`name`*: the overlap layer boundary has not been fully enclosed by the prBoundary. Check that the correct layers have been specified in the *Using geometry on layers* field.

Description: There is geometry on the overlap layer which extends beyond the place-and-route boundary.

Troubleshooting: Check the layers specified in the *Using geometry on layers* field for errors.

ABS-14016

Message: Cell *`name`*: no overlap layer shapes have been generated. Check that the layers have been specified in the *Using geometry on layers* field.

Troubleshooting: For information about specifying overlap layers, see Specifying Layers and Geometries for the Overlap Boundary.

ABS-14017

© 2020 All Rights Reserved.

Message: Cell *name*: has more than one overlap shape generated. Check the layers specified in the *Using geometry on layers* field and the abstract view.

Description: The geometry you supplied in the *Using geometry on layers* field in the *Overlap* tab has led to the tool creating more than one shape for the overlap layer.

This error might also appear if the shapes to be considered for creating the overlap boundary exist on the `boundary` purpose, whereas only the layer(s) is specified in the *Using geometry on layers* field.

Troubleshooting: Set the *Smooth factor to apply* to be greater than the distance between the shapes on the overlap layer and rerun Abstract Generator. This forces the tool to create a single shape.

For information about specifying overlap layers, see Specifying Layers and Geometries for the Overlap Boundary.

To include shapes on the ignored `boundary` purpose, do one of the following:

- Explicitly include the purpose in the geometry specification in the *Using geometry on layers* field.

  For example, if you want the `boundary` shapes on `M1` to be included in the overlap boundary creation, specify the following in the field instead of just `M1`:

  ```
  (M1 boundary) M1
  ```

- Remove `boundary` from the Ignored Purpose(s) field on the *General* tab (*File – General options*).

  Since this is a global , updating the Ignored Purpose(s) field in this way will affect all layer definitions. Therefore, choose this alternative only after due consideration.

ABS-14019

Message: Cell *name*: net *name* has preset pin geometry which violates minimum area rules. You may want to option re-run the Extract step after ensuring that the *Extract signal nets* and/or *Extract power nets* options are turned on to create the correct geometry.

© 2020 All Rights Reserved.

Description: Because full geometry is not being performed, Abstract Generator has created a pin that does not satisfy the minimum area rules set for the layer in the technology file.

Troubleshooting: For information about extracting full geometry, see Extracting Signal Nets and Extracting Power Nets.

ABS-14020

Message: Cell `name`: net `name` has boundary pin geometry which violates the minimum area rules. Although this is only a warning, you may want to inactivate the *Create boundary pins* .

Description: The *Create boundary pins* option has created pins that do not satisfy the minimum area rules in the technology file.

ABS-14021

Message: Cell `name`: net `name` has full shape pin geometry that violates the minimum area rules set in the technology file.

Description: The pin geometry for a given layer must satisfy the minimum area rules defined for that layer in the technology file. If full geometry extraction is performed, and the pin still does not satisfy minimum area rules, it indicates a significant problem in the layout design.

ABS-14022

Message: Cell `name`: geometry has been found abutting or extending past the prBoundary. As a result, the *Create boundary pins* option has been disabled.

Description: Because there is some geometry other than labels, power shapes, or abutments which extends beyond the place-and-route boundary, the *Create boundary pins* option has been disabled automatically. This indicates that there is a problem with the layout design.

ABS-14029

© 2020 All Rights Reserved.

Message: Cell *name*: Trying to create a blockage on layer *layerName*. This layer does not have boundary purpose in the technology data.

Description: This error message is issued when you try to add blockage(s) on a layer that does not have a purpose defined in the technology file.

Troubleshooting: Use the Technology File Manager to define the missing LPPs in the `techLayerPurposePriorities` section of the technology file. Also see ABS-10050.

ABS-14030

Message: Cell *name*: An internal error occurred while performing metal density calculation. *Additional information*.

Troubleshooting: This is an uncommon error message. Look for warning messages received prior to recieving this error message and try to resolve the problem. If you get stuck, contact Cadence support with the problem testcase.

ABS-14031

Message: Cell *name*: An internal problem occurred while performing metal density calculation for the *layer name*. *Additional information*.

Troubleshooting: This is an uncommon error message. Look for warning messages received prior to recieving this error message and try to resolve the problem. If you get stuck, contact Cadence support with the problem testcase.

ABS-14032

Message: Cell *name*: Density data will not be calculated for the non-metal layer *layer name*.

© 2020 All Rights Reserved.

Description: This warning message appears if you are using the *Layer Assignment for Metal Density Regions* table on the *Density* tab for density calculation and have specified a non-metal layer in the table. Abstract Generator will ignore the non-metal layer for density calculation.

ABS-14033

Message: Cell `name`: You are calculating metal density for the `layer name` layer. The technology file does not have a minimum density rule for this layer.

Description: This warning message appears when you try to calculate density for a metal layer for which no minimum density rule has been defined in the technology file. In the absence of the minimum density rule for the metal layer, the density information generated would not be useful.

To add minimum density rule for a metal layer in the technology file, see *Virtuoso Technology Data ASCII Files Reference*.

ABS-14035

Message: Cell `name`: PR boundary is not rectilinear. Edges at non-orthogonal angles will be approximated to larger rectangular shapes, which might overlap other such rectangular shapes. Stair step approximation is currently not supported.

Description: Metal density calculations are based on the dimensions of rectangular density windows. However, when the edges of a PR boundary are non-orthogonal, Abstract Generator will extrapolate the shape to the closest possible rectangle. The resulting rectangular shape encloses regions outside the PR boundary and can overlap other such approximated rectangles.

Troubleshooting: Check the generated abstract to ensure that it meets the required guidelines.

ABS-14036

Message: Cell `name`: does not have a valid PR boundary. The bounding box of the cell view will be used to generate density window(s).

© 2020 All Rights Reserved.

Description: Reconsider the values selected for the *Create boundary* and *Create overlap boundary* options on the Boundary tab (Pins step) and Overlap tab (Abstract step), respectively, to obtain a valid PR boundary for the cell.

ABS-14037

Message: Cell `name`: Site pattern for the gate-array cell `name` cannot be calculated because it contains an instance of the Pcell `name`. Site pattern calculation for Pcells is not supported.

Description: Site pattern calculation for Pcells is not supported.

ABS-14038

Message: Cell `name`: Site pattern for the gate-array cell `name` cannot be calculated. The gate-array cell contains an instance of the sub-cellview `name`, which itself is associated with a site array.

Description: The gate-array cell contains an instance of the sub-cellview name, which itself is associated with a site array.

ABS-14039

Message: Cell `name`: The sub-cellview `name` is mapped to the site `name` in the *Sub-Cellview to Site Mapping* table. However, this sub-cellview is already option attached to the site `name`. The mapping entry specified in the table will be ignored.

Description: The sub-cellview's association to a site is different from what is specified in the *Sub-Cellview to Site Mapping* table on the *Site* tab. Therefore, the mapping entry specified in the table will be ignored.

ABS-14040

© 2020 All Rights Reserved.

Message: Cell `name`: Site pattern for the gate-array cell `name` cannot be calculated. The gate-array cell contains an instance of the sub-cellview `name`, which is not mapped to a site in the *Sub-Cellview to Site Mapping* table. Ensure that the sub-cellview to site mapping is defined in the *Sub-Cellview to Site Mapping* table.

Troubleshooting: Ensure that the sub-cellview is mapped to a site in the *Sub-Cellview to Site Mapping* table on the *Site* tab.

ABS-14041

Message: Cell `name`: Site pattern for the gate-array cell `name` cannot be calculated. The site `name` mapped to the gate-array cell `name` in the *Sub-Cellview to Site Mapping* table does not exist. Ensure that the site to which the sub-cellview is mapped is either predefined or already exists.

Troubleshooting: Ensure that the site to which the sub-cellview is mapped in the *Sub-Cellview to Site Mapping* table exists.

© 2020 All Rights Reserved.

ABS-16000 to ABS-16199: Import/Export Messages

ABS-16001

Message: Cell `name`: The *Import - Logical* command failed to create a cellview. This is because no view name has been specified for importing the logical information (Verilog). To enable the abstract generator to successfully import the verilog file and create the cellview, specify a view name in the *Logical View Name* field in the *Views* tab of the *General options* form. Ensure that the view name (logical) is compatible with the type of logical data (verilog) being imported and then try again.

Description: This error message is issued when Abstract Generator is unable to import the verilog file.

Troubleshooting: Check the verilog file to ensure that its syntax is correct.

ABS-16002

Message: Failed to import the logical information because the `name` logical view name is invalid. Change the *Logical view name* option to a valid name in the *General options* form and then try again.

Description: In the *File – General options* dialog, you have specified a logical view name which is incompatible with the type of logical file you are trying to import. For example, the option might be set to "`schematic`" and you are trying to import a Verilog file.

Troubleshooting: Ensure that the global logical file name is compatible with the file type.

ABS-16100

Message: Bin `name`: Export LEF failed because there are no valid abstracts or there were failures with the tech lef or technology file could not be opened. Ensure that you have the required permissions for the technology file and then try again.

© 2020 All Rights Reserved.

Description: If the problem is related to technology, bin, or cell failures, then it is likely that difficulties were encountered during the `lefout` translation process.

However, the problem might be related to incorrect permission settings on the technology file.

ABS-16005

Message: Failed to import the technology LEF because the current attached technology library is the default *`name`*. Attach a non-default technology library to merge LEF technology information and then try again.

Description: It might be that you created a new library using *File – Library – New* and did not specifically choose to attach a particular technology library (using *File – Library – Attach Technology*) and `cdsDefTechLib` was added by default.

ABS-16006

Message: Failed to import the technology LEF because the *display.drf* file is not writable. Ensure that the *display.drf* is writable and then try again.

Troubleshooting: Ensure that the permissions on the `display.drf` file are set correctly so that the technology LEF can be imported.

ABS-16103

Message: LEFOUT Failed. Either no valid abstract views are present in the library or the cells given in the celllist file are not found in the abstract with valid views.

Troubleshooting: Ensure that the cells specified in the cell list file exist in the library.

ABS-16104

Message: LEFOUT Failed. File name containing the list of cells is invalid or is not found in the given path.

© 2020 All Rights Reserved.

Troubleshooting: Ensure that the correct path to the cell list file has been specified and that you have read permission to the file.

© 2020 All Rights Reserved.

ABS-17000 to ABS-17099: Messages

ABS-17000

Message: Failed to import *Name* option because the option has not been specified. Specify the option and then try again.

Description: The option given has not been set but is required for the operation you want to perform.

Troubleshooting: Set the options in question and perform the action again.

ABS-17003

Message: The *Name* global option has an invalid value and has been reinitialized from *value* to *value*.

Description: When there is a change in the technology file that is being used any invalid global options must be reinitialized.

ABS-17004

Message: The *name* bin's *Name* option has an invalid value and has been reinitialized from *value* to *value*.

Description: When there is a change in the technology file that is being used any invalid bin options must be reinitialized.

ABS-17006

Message: The entry in the *Text Manipulation* table for bin *name* which maps '*string*' to '*string*' has been changed to map '*string*' to '*string*' in order that pin names are created with angle brackets rather than square brackets. Re-import the logical view and rerun the Pins step to ensure that the generated abstract views are correct.

© 2020 All Rights Reserved.

Description: Previously, bus characters were represented using square brackets; now they are represented using angle brackets. If Abstract Generator finds any entries that map to square brackets, it changes them as follows.

- Entries mapping `( )` to `[ ]` are deleted.
- Entries mapping `< >` to `[ ]` are changed to map `[ ]` to `< >`.

ABS-17007

Message: Detected an entry in the *Text Manipulation* table for bin *`name`* which may be mapping angle brackets or parentheses to square brackets (maps '*`string`*' to '*`string`*'). From release 5.0.33.9, bus characters are represented internally as angle brackets, rather than square brackets. The entries in the *Text Manipulation* table should map square brackets to angle brackets (e.g., '\\[([0-9]+)\\]' to '<\\1>'). Once this option has been set up correctly, re-import the logical view and rerun the Pins step in order to ensure the generated abstract views are correct.

Description: The internal representation of bus characters was changed from the stated release onwards. Previously they were represented using square brackets; now they are represented using angle brackets. Abstract Generator has found an entry in the Text Manipulation table which might be mapping angle brackets to parentheses or square brackets.

Troubleshooting: Check the entry and correct it as required before proceeding.

ABS-17008

Message: The Global option ImportLogicalType has been assigned to an invalid value TLF

Description: The valid values for the `ImportLogicalType` global option are `LIB` and `Verilog`.

© 2020 All Rights Reserved.

ABS-11900 to ABS-11999: System Messages

ABS-11904

Message: Failed to delete the system bin because system bins such as *Core*, *IO*, *Corner*, *Block*, and *Ignore* cannot be deleted.

Description: The five system bins are `Core`, `IO`, `Corner`, `Block`, and `Ignore`. You cannot delete or rename them. For more information about system bins, see Bin Pane.

ABS-11905

Message: Failed to delete the *`name`* user bin because the bin contains cells. Transfer all cells to another bin and then try again.

Troubleshooting: For more information about bins, see Bin Pane.

ABS-11906

Message: Failed to open the *`library name`* library because the library is not listed in the *cds.lib* file. Ensure that the library is defined in the *cds.lib* file and then try again. If you are using a replay file, check the name and location of the library in *cds.lib* to ensure that the library exists.

Description: This error message is generated when you try to open a library that is not listed in the `cds.lib` file.

Troubleshooting: If you are using a replay file, check the name and location of the library in the `cds.lib` file to ensure that it exists. You could also verify if the library defined in the `cds.lib` file exists.

ABS-11907

Message: Failed to open the *`name`* bin because either the bin does not exist or you are using an options or replay file from another library and the bin does not exist in the destination library. Ensure that the bin exists and then try again.

© 2020 All Rights Reserved.

Description: This error message is generated when you try to run a command on a bin that does not exist. A common source of this error is using an options file or replay file copied from another library and the bin not existing in the destination library.

Troubleshooting: Check that the name of the bin is correct.

ABS-11908

Message: Failed to open the `name` cell because either the cell does not exist or you are using an options or replay file from another library and the cell does not exist in the destination library. Ensure that the cell exists and then try again.

Description: This error message is generated when you try to run a command on a cell that does not exist. A common source of this error is using an options file or replay file copied from another library and the cell not existing in the destination library.

Troubleshooting: Check that the name of the cell is correct.

ABS-11909

Message: Failed to find the `name` view. Check the view names in the *Views* tab of the *General options* form.

Troubleshooting: For information about current view name settings, see Views Tab.

ABS-11910

Message: Failed to change the value of the `name` terminal because the terminal does not exist. Ensure that the terminal name that you have specified is correct and then try again.

Description: This error message is generated when you try to change the value of a terminal that does not exist. A common source of this error is using an options file or replay file copied from another library and the terminal not existing in the destination library.

Troubleshooting: Check that the name of the terminal is correct.

© 2020 All Rights Reserved.

ABS-11911

Message: Failed to change the value of *name* global option because either the option does not exist in the current version of Abstract Generator or you have incorrectly edited the options or replay file. Check if the option exists in the current version of Abstract Generator and if you have edited the options or the replay file correctly and then try again.

Description: This error message is generated when you try to change the value of a global option that does not exist. A common source of this error is using an options file from an older version of Abstract Generator whereas the option might have been removed in the newer version.

Another source of this error could be the manual editing of the options or replay file leading to a typo in the name of the global .

ABS-11912

Message: Failed to change the value of *name* bin option because either the option does not exist in the current version of Abstract Generator or you have incorrectly edited the options or replay file. Check if the option exists in the current version of Abstract Generator and if you have edited the options or the replay file correctly and then try again.

Description: This error message is generated when you try to change the value of a bin option that does not exist. A common source of this error is using an options file from an older version of Abstract Generator whereas the option might have been removed in the newer version.

Another source of this error could be the manual editing of the options or replay file leading to a typo in the name of the bin .

ABS-11913

Message: Failed to set the property *name* because it is an invalid property. Check the property you have specified in the options file, replay file, or the abstract command line and then try again.

© 2020 All Rights Reserved.

Description: This error message is generated when you try to set an invalid property.

A common source of this error could be the manual editing of the options or replay file leading to a typo in the name of the property.

ABS-17008

Message: *binName Name* invalid value *value*.

Description: This error message appears when you specify an invalid value for an option of the `absSetBinOption` function. A corresponding information message displays after the error, which lists the valid values for the option specified as an argument to the function.

Troubleshooting For information about the options for the `absSetBinOption` function and their valid values, see `absSetBinOption`.

ABS-11914

Message: settings for a bin are being copied to another type of bin. The tool uses different options settings based on the type of bin to aid abstract generation.

Description: This error message appears when you click the *Copy From Bin* button on the *Bin options* dialog box (*Bins – options*) to copy options from one bin to the selected bin of a different type. Default options are set on bins to aid in the process of abstract generation, depending on the type of the bin.

ABS-17011

Message: *name* option has been removed from this version of the Abstract Generator.

Description: This error message appears if you try to set one of the wide wire options, `AbstractBlockageModelWideObs` or `AbstractBlockageWideObsMinWidth`. This is because from release IC6.1.0 and onwards, the options for modeling wide wire obstructions have been removed.

© 2020 All Rights Reserved.

Troubleshooting: Use the options *Pin Cutout*, *Max Space*, and *Corridor Cut* for appropriately modeling cover, detailed, and shrink blockages.

ABS-11915

Message: The abstract generator failed to create the new bin. A bin with the name `binName` already exists. Use a different name to create the bin.

ABS-19029

Message: Replay of `name` file aborted at line `number` because of errors.

Description: A replay file has caused an error, such as loading a library that does not exist.

ABS-19031

Message: Abstract Generator failed to open the cellview because layout editor is not specified in the PATH. Ensure that layout editor is specified correctly in the PATH and then try again.

Description: This error message appears when you are unable to open a cellview from Abstract Generator. The possible reason could be that the path has not been correctly set in your environment to include layout.

Troubleshooting: Check the path and the current Cadence hierarchy.

© 2020 All Rights Reserved.

ABS-11500 to ABS-11599: OAToolbox Messages

ABS-11500

Message: Cell *name*: Row *name* in view *viewName* has a missing property *propertyName* and will not be converted.

ABS-11501

Message: Cell *name*: Row *name* in view *viewName* references a site def *siteDefName* which has not been defined in the technology database.

ABS-11502

Message: Couldn't open master *libName*, *cellName*, *viewName*, referenced in cellview *libName*, *cellName*, *viewName*.

© 2020 All Rights Reserved.

ABS-20000 to ABS-20099

ABS-20007

Message:	There are no vias specified in the technology file. Ensure that the selected constraint group has a *ValidVias* section or at least one via is specified in the *standardViaDefs* subsection of the *viaDefs* section in the technology file. In addition, ensure that the *function* argument is set to '*cut*' for the via in the *functions* subsection of the *layerRules* section.
Type:	Warning
Description:	The `standardViaDefs` section of the technology file defines the via layers. Because a design requires at least two metal layers, there must be at least one via layer defined to connect them. Abstract Generator creates extraction connectivity list using `validvias` section.

© 2020 All Rights Reserved.

ABS-23000 to ABS-23099

ABS-23005

Message: Cell *name*: Terminal *name* has been ignored although it has the mustJoin attribute because it was not indicated in the Create Must Join Pins If Required section on the Signal page. To ensure that must-join pins are created for terminals, select the Always option or include specific terminals in the Only on terminals named field.

© 2020 All Rights Reserved.

F

Tutorial: Generate Abstracts by Using the Standalone Abstract Generator

This tutorial explains the methods involved in generating abstract LEF from Virtuoso design environment data for use by place-and-route tools such as Innovus™ Digital Implementation System (Innovus). The tutorial will guide you through a typical abstract generation flow. It does not explain full Abstract Generator functionality, so if you come across any issues that you cannot resolve, contact your local Cadence Application Engineer or Cadence Customer Support directly.

Note: Cadence Customer Support is region specific. To find out the e-mail address, phone number, or fax number for your region. Select the *Contacts* button from the Cadence Online Support home page.

Sample Library Data

The sample library material used in this tutorial can be found at the location below.

`tools/dfII/samples/tutorials/abstract`

This path will be referred to as `<tutorial_path>`.

License Requirement

You should have at least one license of Virtuoso® Layout Suite XL.

Virtuoso Design Environment to LEF Flow Overview

The process of generating abstract LEF from the Virtuoso design environment by using the standalone Abstract Generator can be summarized by the following procedures:

1. **Opening a Cell library and Checking Technology Data**

© 2020 All Rights Reserved.

In this step, we will launch Abstract Generator and open a library. At this stage, we will also check the technology data.

Abstract Generator can open libraries existing in the Virtuoso design environment, referenced in your `cds.lib` file, which would bypass the need for this step.

2. **Generating Abstract Views**

   Once we have a library with technology information and cell data, we can then begin to generate abstracts from the cell layout view.

3. **Exporting Abstract LEF**

   The final step is to generate abstract LEF from the cell abstracts.

   Abstract Generator supports mapping DFII cellType values to the LEF MACRO and LEF58_CLASS properties. These properties are mutually exclusive. For details about LEF58_CLASS, see *LEF/DEF 5.8 Language Reference*.

© 2020 All Rights Reserved.

Opening a Cell Library and Checking Technology Data

Starting Abstract Generator

1. Change the current path to where the tutorial designs are located:

   ```
   cd <tutorial_dir>
   ```

   Remember, `<tutorial_dir>` is the equivalent of `tools/dfII/samples/tutorials/abstract` in the Virtuoso design environment installation hierarchy.

2. Launch Abstract Generator:

   ```
   abstract
   ```

   The Abstract Generator main window appears:

© 2020 All Rights Reserved.

Opening the Library

In this step we will open the `amspLL` library which contains our tutorial data.

1. Select *File – Library – Open* to display the Open Library form:

2. Select the `amsPLL` library and click the *OK* button.

   The library will be opened. At this point the technology data is verified. Any errors would be reported here. However, in this example no errors should be reported.

   There are 17 cells in the *Core* bin and 11 cells in the *Ignore* bin. The top-level cell we are interested in is called `pllclk` in the *Core* bin. This cell is a block so we will now want to move it to the *Block* bin for abstract generation. We do this because Abstract Generator sets default options depending on the bin a cell/block is located in.

3. Select the `pllclk` cell in the *Core* bin of the cell pane, then select *Cells - Move*.

4. In the *Move Selected Cells* dialog box, select *Block* and click *OK*. The `pllclk` cell is moved to the *Block* bin.

5. In the *Bin* pane, select *Block.* The contents of the *Block* bin will be displayed to show the selected `pllclk` cell.

   We are now ready to start abstract generation.

   Note: You can inspect the layout view for the `pllclk` cell by selecting *Cells – Edit – Layout*.

© 2020 All Rights Reserved.

Generating Abstract Views

With the layout and logical information successfully imported into Abstract Generator, we can now begin to generate an abstract view for the `pllclk` block.

Running the Pins Step

1. With the `pllclk` block selected, choose *Flow – Pins* to display the *Running step Pins* form. The *Map* tab is the first tab on view. There are existing terminals in the layout so we are not required to set any options on this tab.
2. Click the *Run* button at the bottom of the form to create the pins view for the `pllclk` block. An orange exclamation mark will be placed against the `pllclk` block under the *Pins* column in the cell pane. This informs us that the step has completed, but with warnings. In this example, these can however safely be ignored.

   Note: You can check the successful creation of the Pins view by selecting *Cells – Edit – Pins*. The Virtuoso layout editor is opened with the Pins view of the `pllclk` block on display.

Running the Extract Step

We shall now create the extraction view for the `pllclk` block. For the signal pins we will maintain the pins identified by the Pins step, but for the power pins we will run connectivity extraction. This is done to make more of the power routing on the block available as pins for connecting at the next level.

1. Select *Flow – Extract* to display the *Running step Extract* form.
2. Click the *Antenna* tab and ensure that the following process antenna calculations have been switched *on*:

   Calculate input pin antenna
   Calculate output pin antenna

© 2020 All Rights Reserved.

Calculate inout pin antenna
Calculate antenna metal area

3. In the *Layer Assignment for Antenna regions* table, we need to make sure that gate and drain regions can be recognized. Ensure that `Poly` is listed in the *Layer* column and that its *Geometry Specification* is set to `Poly and Oxide` and *Region* is set to *Gate*. Ensure also that `Oxide` is listed in the *Layer* column and that its *Geometry Specification* is set to `Oxide andnot Poly` and *Region* is set to *Drain*.

4. As we are not going to extract signal nets, ensure that the *Use different layer assignments for antenna calculations only* option is selected.

5. All metal and via layers should already appear in the *Layer Assignment for Antenna Extraction* table.

6. Ensure that the table contains entries for `Cont`, `Poly` and `Oxide` layers and the corresponding entries in the *Geometry Specification* column. These entries are required in the table to run antenna calculation. Ensure that the g-spec specified for the `Oxide` layer is `Oxide andnot Poly`. This will ensure that `Oxide` is correctly extracted.

7. Click the *General* tab. In the *Layer connectivity* field, you will see a list of triple layers.

© 2020 All Rights Reserved.

A triple layer indicates that the first two layers in the triple are connected by the third (a via or a contact). Ensure that the metal layers are all present and that there are also triple layers specifying `Poly` to `Metal1` and `Oxide` to `Metal1` connectivity.

8. If the `Poly` and `Oxide` triple layers have not been defined, add the following to the end of the triple layer list:

   ```
   (Poly Metal1 Cont) (Oxide Metal1 Cont).
   ```

9. Click the *Signal* tab. Ensure that the *Extract signal nets* option is not selected. The existing pin shapes are sufficient so there is no requirement for signal net extraction.

10. The final task, before running the Extract step, is to set the power net extraction options. These settings are block dependent. For the `pllclk` block, the Pins step created power pins on `Metal3` along the bottom edge of the block. The `Metal4` vertical segments on the right edge of the block should also be modelled as power pins.

11. In the *Power* tab, ensure that the *Extract power nets* option is selected. In the *Layer Assignment for Power Extraction* table, ensure that *Create Pins* is not selected for `Metal1` and `Metal2`, and also for `Via1`,`Via2`, `Via3`, `Via4` and `Via5`.

12. Click the *Run* button at the bottom of the form to create the Extract view for the `pllclk` block.

    Note: This process may take a couple of minutes.

    A green tick will be placed against the `pllclk` block under the *Extract* column in the cell pane.

© 2020 All Rights Reserved.

Running the Abstract Step

Next, we will create the Abstract view for the `pllclk` block.

1. Select *Flow – Abstract* to display the *Running step Abstract* form. The *Adjust* tab is the first tab on view.
2. Select the *Create boundary pins* option for *Signal Nets*.

   In the extraction step, we did not run signal net extraction, so this should not strictly be required because the pins that were mapped in the Pins step are usually fine. However, if the pins created in the Pins step extend deep into the `pllclk` block, perhaps due to text labels being mapped onto an underneath path, then it is necessary to turn these larger pins into small square pins at the boundary of the block. We do this by selecting the *Create boundary pins* .

   For *Power Nets*, the power pins in the `pllclk` block are neither boundary pins nor ring pins, therefore, both the *Create boundary pins* and *Create ring pins* options should not be selected.

© 2020 All Rights Reserved.

3. Click the *Blockage* tab and ensure that each metal layer is listed in the *Layer Assignment for Blockages* table and that their *Blockage* type set to *Cover*.

4. To create a better Abstract view, we will shrink the cover blockage away from the cell boundary to ensure that `Metal3` and `Metal4` power pins are exposed. To achieve this, scroll to the right in the *Layer Assignment for Blockages* table to the *Boundary* column and enter *15* (microns) for *Layers* `Metal3`, `Metal4`, `Metal5`, and `Metal6`.

5. Click the *Grids* tab and set *Grid analysis mode* to *off*. This should be the default for the *Block* bin.

6. Click the *Run* button at the bottom of the form to create the Abstract view for the `pllclk` block.

7. A green tick will be placed against the `pllclk` block under the *Abstract* column in the cell pane. Again, the newly created Abstract view can be inspected in the Virtuoso layout editor by selecting *Cells – Edit – Abstract*.

© 2020 All Rights Reserved.

Exporting Abstract LEF

The final task in completing the generation of abstract LEF from Virtuoso design environment is to export the LEF from Abstract Generator.

1. Select *File – Export – LEF*. The Export LEF form is displayed. By default, all information is written to the LEF file.
2. However, as we already have a technology LEF file for this tutorial, uncheck the *Export Tech LEF Data* checkbox to prevent technology LEF being written to the LEF file.

3. Ensure that the *LEF Filename* option is set to `abstract.lef`.
4. Change the *LEF Version* to be 5.5.
5. Click the *OK* button to export the abstract LEF for the `pllclk` block.

   The `abstract.lef` file will be created in the Abstract Generator *run* directory (by default `abstract`), which was specified when Abstract Generator was first invoked from the executable in the Virtuoso design environment.
6. With the LEF file now successfully exported, close Abstract Generator by selecting *File – Exit*.

© 2020 All Rights Reserved.

The same Abstract Generator run directory can be used again for future Abstract Generator sessions. You can also re-invoke the abstract executable again from the same run directory.

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

G

Technology File Requirements for Abstract Generation

This appendix describes how Abstract Generator handles technology information in the Virtuoso design environment, outlines the technology information required to run Abstract Generator on OpenAccess, identifies potential technology file problems, and suggests possible solutions to those problems.

It covers the following topics:

- Handling Technology Information in the Virtuoso Design Environment, which describes how to generate technology information for Abstract Generator, how to supplement that information by importing LEF, and how to modify technology information using the Technology File Manager.
- Technology File Requirements, which lists the technology information required to run Abstract Generator on OpenAccess. This includes the definition of layers and layer rules, layer functions, physical rules, and vias along with specific requirements for antenna calculation, and viewing data in the Virtuoso Layout Editor.
- Potential Issues Related to the Technology File in the Virtuoso Design Environment, which lists common problems found in technology files in the Virtuoso design environment and provides information about how to fix these to enable Abstract Generator to run.

This appendix also describes how to convert Abstract Generator replay files from common-lisp format into SKILL or Tcl format in the section Converting Common Lisp Files to SKILL or Tcl Format.

Additional Information

- For information about the technology information messages issued by Abstract Generator, see ABS-10000 to ABS-10099: Technology Messages.
- For information about the Technology File Manager, technology files, and their construction, see *Virtuoso Technology Data ASCII Files Reference*.

© 2020 All Rights Reserved.

Handling Technology Information in the Virtuoso Design Environment

Technology information provides details of the process technology used during IC fabrication, including the names of layers, colors and fill patterns, GDSII layer mapping data, and design rules for various layers and vias.

This section describes how Abstract Generator handles OpenAccess technology information associated with your design library.

Technology Information in the Virtuoso Design Environment

The following are the three main sources of technology information for a design library in the Virtuoso design environment:

- A binary technology file, called `tech.db`, stored in the library directory
- A separate technology library, which is attached as a property to the design library
- A display resource file, `display.drf`, stored in the library directory

Creating a New Library

If you use *File – Library – New* to create a new library, Abstract Generator automatically attaches the Cadence default technology file shipped with the software, `cdsDefTechLib`, to the library.

The default technology file is unlikely to contain enough information to satisfy the internal technology checker of Abstract Generator, so you must always replace or supplement the technology information before you begin abstract generation.

There are two ways of doing this using the Abstract Generator GUI:

- Select *File – Library – Attach Technology* to attach an existing technology file to be used with your new library.

Important

During abstract generation, the tool might need to update your technology information; for example, the tool might update the technology file with information about sites and layer-purpose pairs. You must ensure that an attached technology file has permissions that allow it to be updated by Abstract Generator, otherwise you will need to add the missing information manually.

© 2020 All Rights Reserved.

- Use *File – Import – LEF* to import a LEF technology file. Abstract Generator automatically converts the information in the LEF file into appropriate technology information for OpenAccess.

Importing an Existing Library

If you are importing an existing library into Abstract Generator and the library has its own technology file (`tech.db`), Abstract Generator uses that file as the main source of technology information.

If the existing library has a different technology library attached, Abstract Generator uses that instead. You can use *File – Library – Attach Technology* to attach a different technology file to your library.

Tip

If your library has an attached technology file, Cadence recommends that you move `tech.db` file out of the library directory before you start generating abstracts so that possible conflicts can be avoided.

Modifying Technology Information

It is likely that you will need to modify the technology information for your library to provide all the information Abstract Generator needs.

You can modify the technology information in the following two ways:

- Use *File – Import – LEF* to import additional information into your existing technology file. Any additional information present in the LEF file is added to the existing technology information for the library.

Tip

If you want to replace the existing technology information with the information from a LEF file, select the *Overwrite* option on the *Import LEF* form.

- If you need to make more extensive manual changes to your technology information, use *Technology File Manager* to generate an ASCII version of the technology file. You can edit this file as required and recompile it for use with the library.

  To launch the *Technology File Tool Box*, select *Tools – Technology File Manager* in the CIW.

© 2020 All Rights Reserved.

For more information about the Technology File Manager and technology files, see the *Virtuoso Technology Data User Guide*.

Example

An example of a situation in which the technology file might need to be updated is when purposes are missing for a routing layer. This is because the technology file must define the LPPs for all routing layers. In case purposes are not defined for a routing layer, Abstract Generator updates the technology file by automatically creating the LPPs. Alternatively, you can update the technology file manually by using the Technology File Manager. Abstract Generator will not proceed if an LPP is missing for a routing layer (ABS-10034).

In both cases where either you update the technology file or Abstract Generator does it automatically, the technology file should be writable (ABS-10050).

Important

Abstract Generator allows you to open a library and proceed with the Abstract Generator flow if the pin and net are not defined for masterslice or cut layers. Abstract Generator issues warning messages for the missing purposes on the masterslice or cut layers. The drawing purpose does need to be defined for routing as well as masterslice and cut layers. However, if during the abstract generation flow, a pin or a blockage is created on a masterslice or cut layer that does not have a corresponding purpose defined in the technology file, Abstract Generator generates one of the following errors:

ABS-12024: `Cell name: Trying to create a pin shape on layer layerName. This layer does not have pin purpose in the technology data.`

ABS-13022: `Cell name: Trying to create a pin shape on layer layerName. This layer does not have net purpose in the technology data.`

ABS-14029: `Cell name: Trying to create a blockage on layer layerName. This layer does not have boundary purpose in the technology data.`

Technology File Requirements

This section describes the technology information required for Abstract Generator to run in the Virtuoso design environment.

- Layer Definitions

© 2020 All Rights Reserved.

- Via Definitions
- Physical Rules
- Devices
- Voltage Dependent and Layer-Purpose-Based Rules

This section also lists specific requirements related to the particular tasks that you might want to perform by using Abstract Generator.

- Specific Requirements for Antenna Calculation
- Specific Requirements for Metal Density Calculation
- Specific Technology Requirements for Viewing Data In Layout
- Specific Requirements for Specifying Place-and-Route Information

Layer Definitions

The table below lists the required subsections from the `layerDefinitions` section.

Subsection	Description	Example

`techLayers`	Defines the layers to be used in your design.
	
	This layer list must include two or more metal layers and one or more via layers. Each layer must have a name, a number, and an abbreviation for the layer name.
		`layerDefinitions(`
		`techLayers(`
		
		`;( LayerName Layer# Abbreviation )`
		`;( --------- ------ ------------ )`
		`; User-Defined Layers:`
		`. .`
		`( ndiff 1 ND )`
		`( pdiff 2 PD )`
		`( pwell 13 PW )`
		`. .`
		`) ;techLayers`
		`) ;layerDefinitions`

© 2020 All Rights Reserved.

Subsection	Description	Example

`techLayerPurpose Priorities`	Defines the display priority order of layer purpose pairs (LPPs).
	
	Lists the layer-purpose pairs (LPPs) in the order in which they are to be displayed; applications display an LPP on top of the one that precedes it in the list and underneath the one that follows it.
	
	All routing layers must have the purposes net, pin, and drawing. Abstract Generator adds these automatically if required.
		`layerDefinitions(`
		`  techLayerPurposePriorities(`
		
		`  ;layers are ordered from the lowest`
		`  ;to the highest priority`
		
		`  ;( LayerName  Purpose  )`
		`  ;( ---------  --------  )`
		`  ( nwell  drawing  )`
		`  ( nwell  pin  )`
		`  ( nwell  net  )`
		`  ( nwell  boundary  )`
		
		`  ) ;techLayerPurposePriorities`
		`) ;layerDefinitions`

Via Definitions

A via layer connects two conducting layers.

On OpenAccess 2.2, information about the via layers is determined from the `viaDefs` section of the technology file. There are two types of `viaDefs` — `standardViaDefs` and `customViaDefs`.

Standard Vias

The `standardViaDefs` has a unique name and is associated with two layers and a list of via parameters with default values.

Example

```
viaDefs(
 standardViaDefs(

 ;(viaDefName layer1 layer2 (cutLayer cutWidth cutHeight [resistancePerCut])
 ;(cutRows cutCol (cutSpace))
 ;(layer1Enc) (layer2Enc) (layer1Offset) (layer2Offset) (origOffset)
 ;[implant1 (implant1Enc) [implant2 (implant2Enc)]])
```


© 2020 All Rights Reserved.

```
 ;(-----------------------------------------------------------------------)

  ( via1Array METAL1 METAL2 ("VIA12"   0.36     0.36)
  (1 1     (0.36    0.36))
  (0.0 0.0)   (0.0 0.0)    (0.0 0.0)     (0.0 0.0)       (0.0 0.0)
  )
 ) ;standardViaDefs
) ;viaDefs
```

Custom Vias

The `customViaDefs` has a unique name and is associated with a master cellview and two layers. A `customViaDef` is a special purpose via. To specify a custom via, the cellview on which it is based must exist in the specified library.

Example

```
viaDefs(
 customViaDefs(
 ;(viaDefName libName cellName viewName layer1     layer2  resistancePerCut)
 ;(---------- ------- -------- -------- ------     ------  ----------------)

  ( M2_M1_Via1 lib     M2_M1    via      METAL1     METAL2  4.0 )
  ( M1_Poly    lib     M1_Poly  via      POLY       METAL1  4.0 )
 ) ;customViaDefs
) ;viaDefs
```

For more information about `viaDefs`, see *Virtuoso Technology Data ASCII Files Reference*.

Physical Rules

The `spacingRules` subclass included in the `physicalRules` class in the OpenAccess 2.0 technology file is mapped to the `spacings` subsection in the `foundry` constraint group in the OpenAccess 2.2 technology file.

© 2020 All Rights Reserved.

In addition to the `spacings` section, Abstract Generator refers to the `spacingTables` section, also in the `foundry` constraint group.

Constraint Category	Description	Example

`spacings`	Specifies the spacing rules, defining the amount of enclosure required when two layers overlap.
	
	All routing layers must have `minWidth` and `minSpacing` values specified.
	
	The `minWidth` constraint specifies the minimum width for any shape on the layer. The `minSpacing` constraint specifies the minimum spacing between objects on a single layer.
		`constraintGroups(`
		`( "foundry"`
		`;physical constraints`
		`spacings(`
		
		`;( constraint layer1 layer2 value )`
		`;( ---------- ------ ------ ----- )`
		
		`( minWidth "Metal1" 0.3 )`
		`( minSpacing "Metal1" 0.3 )`
		`) ;spacings`
		`) ;foundry`
		`) ;constraintGroups`

`spacingTables`	Defines the lookup table(s) for the voltage-based rules, layer-purpose rules, and layer-spacing rules. Abstract Generator looks up the **`spacingTables`** section for these rules to determine the maximum spacing value to be used for cover blockages and for pin cutouts.	`constraintGroups(`
		`( "foundry"`
		`spacingTables(`
		
		`;( constraint layer1 [layer2]`
		`;((index1Definitions`
		`;[index2Defintions])`
		`;[defaultValue])`
		`;(table) )`
		
		`( minSpacing METAL1`
		`(( "width" nil nil ))`
		`(`
		`0.5 0.333`
		`0.78 0.99`
		`)`
		`) ;spacingTables`
		`) ;foundry`
		`) ;constraintGroups`

© 2020 All Rights Reserved.

Abstract Generator does not recognize spacing rules that are defined using `LEF58` constructs. For example the following is an incorrect definition:

```
PROPERTY LEF58_SPACING "
    SPACING 0.077 NOTCHLENGTH 0.140
CONCAVEENDS 0.037 ;" ;
```

The following definition is correct and is recognized by Abstract Generator:

```
LAYER M1
    TYPE ROUTING ;
    DIRECTION VERTICAL ;
    PITCH 0.090000 0.064000 ;
    OFFSET 0.045000 0.000000 ;
    MINWIDTH 0.032 ;
    MAXWIDTH 2 ;
    WIDTH 0.032 ;
SPACING 0.077 NOTCHLENGTH 0.140 ;

END M1
```

For more information, see *Virtuoso Technology Data ASCII Files Reference*.

Application of the Minimum Area Rule for Pins

In addition to the shape, Abstract Generator respects the `minArea` and `minRectArea` technology file constraints while creating pins. The minArea constraint defines the minimum area for all pins; whereas the `minRectArea` constraint defines the minimum area for only rectangular-shaped pins.

When both constraints are defined, then the `minRectArea` constraint applies to all rectangular pins, whereas the `minArea` constraint applies to all non-rectangular pins.

If the `minRectArea` constraint is not defined, then the `minArea` constraint applies to both, rectangular as well as non-rectangular pins.

Devices

The device definitions in OpenAccess 2.2 are translated to via definitions.

© 2020 All Rights Reserved.

Via Cellviews in OpenAccess

In OpenAccess, the `lef2oa` translator generates a via as a layout cellview with the view name `via` and includes it in the library; For example,

```
library/viaName/via/layout.oa
```

It also adds a corresponding `customViaDef` to the technology file, which refers to the via cellview.

Voltage Dependent and Layer-Purpose-Based Rules

In addition to the layer rules, such as **`minSpacing`** and **`minWidth`**, Abstract Generator now supports the voltage dependent and the layer-purpose-based spacing rules, such as **`minVoltageSpacing`** (voltage dependent rule) and **`minSpacing`** (layer-purpose-based). These spacing rules are defined in the **`spacings`** and **`spacingTables`** section of the technology file. While loading a library, Abstract Generator reads the rules defined in the technology file and use the read rules for generating abstracts.

The option that enables Abstract Generator to read the voltage-based and layer-purpose-based rules is **`ReadVoltageAndPurposeRules`**. To enable reading of rules, this option should be set to **`TRUE`**. By default, this option is set to **`FALSE`**.

```
absSetOption(”ReadVoltageAndPurposeRules” “TRUE/FALSE”)
```

Abstract generator conforms to the voltage and layer-purpose-based spacing rules while it calculates the spacing value for cover blockage and pin cutouts. The spacing value used for cover blockage is the maximum value that is available from the voltage dependent, layer-purpose-based, and layer-based spacing rules. The same maximum value is used to cut the spacing around pins.

An example of a voltage dependent and layer-purpose-based spacing rule is given as follows.

```
spacingTables(
    ;(constraintlayer1layer2
    ;   ((index1Definitions[index2Definitions)[defaultValue])
    ;   (table))
    ;(----------------------------------------)
     (minVoltageSpacing         "Metal1"
      (("voltage" nil  nil))
      (
       2.03.0
       5.03.5
     )
    )
```


© 2020 All Rights Reserved.

```
     (minVoltageSpacing         "Metal2"
      (("voltage" nil  nil))
      (
       2.04.0
       5.04.5
      )
     )
    );spacingTables
spacings(
         ;(minSpacing                ("Metal1" "drawing")3.2)
          (minSpacing                ("Metal1" "pin")    3.4)
          (minSpacing                ("Metal2" "drawing")3.6)
          (minSpacing                ("Metal2" "pin")    3.8)
        );spacings
```

In the above example, the maximum value of the **minVoltageSpacing** rule for Metal1 layer is **3.5**. The **minSpacing** value of Metal1 layer for purpose **drawing** and **pin** is **3.2** and **3.4**, respectively. Therefore, the values that are considered to calculate the **minSpacing** value are **3.2**, **3.4**, and **3.5**. Because **3.5** is the maximum value that is available from the voltage dependent and layer-purpose-based spacing rules for the Metal1 layer, the **minSpacing** value used for cover blockage and pin cutout is **3.5**.

Specific Requirements for Antenna Calculation

This section describes the technology file requirements for using the antenna calculation capabilities of Abstract Generator.

In the technology file on OpenAccess 2.2, the antenna rules have been moved to the `antennaModels` section of the `foundry` constraint group. You specify a separate `antennaModels` group for each oxide type and can specify `antennaModels` for up to four oxide types.

If an `alternate foundry` constraint group (AFCG) is defined in the `.cdsenv` file, Abstract Generator replaces the `foundry` constraint group and uses the specified AFCG instead.

You can specify an AFCG by setting the following environment variable in your `.cdsenv` file:

```
cdba.layout AlternateFoundryCG string "AFCG_name"
```

The constraint specifies the antenna ratios for one oxide type corresponding to a specific thickness of the gate oxide. You can specify the thickness value in the `techLayerProperties` subsection in the `layerDefinitions` section if antenna side area calculation is to be performed.

© 2020 All Rights Reserved.

Note: In OpenAccess, you must specify either side antenna values or non-side antenna values. OpenAccess supports only one of them at a time.

During the Extract step, the following factors are supported while calculating the antenna value. These factors are defined in the `antennaModels` section of the foundry constraint group in the technology file.

- (antennaCutFactor) cumulativeRoutingPlusCut

  If this factor is found in the technology file, the values of the cumulative antenna ratio for `CUT` layers are dumped in the LEF file. In addition, the values of the cumulative antenna ratio for `CUT` layers are added to the cumulative antenna ratio values of the metal layers. This factor has a boolean value. By default, this factor is set to `nil`.

- (antennaMetalFactor) diffArea

  If this factor is found in the technology file, Abstract Generator uses the value of this factor to calculate the cumulative antenna ratio values for the metal layers. However, if this factor is not found in the technology file, Abstract Generator sets it to the default value 1.0 and calculates the cumulative antenna ratio values. This factor has float values.

- (antennaDiffPlusFactor) diffPlus

  If this factor is found in the technology file, Abstract Generator uses the value of this factor to calculate the cumulative antenna ratio values for the metal layers. However, if this factor is not found in the technology file, Abstract Generator sets it to the default value 0.0 and calculates the cumulative antenna ratio values. This factor has float values.

- (antennaDiffMinusFactor)) diffMinus

  If this factor is found in the technology file, Abstract Generator uses the value of this factor to calculate the cumulative antenna ratio values for the metal layers. However, if this factor is not found in the technology file, Abstract Generator sets it to the default value 0.0 and calculates the cumulative antenna ratio values. This factor has float values.

- (antennaDiffAreaReduceFactor) diffAreaReduce

  If this factor is found in the technology file, Abstract Generator uses the value of this factor to calculate the cumulative antenna ratio values for the metal layers. However, if this factor is not found in the technology file, Abstract Generator sets it to the default value 1.0 and calculates the cumulative antenna ratio values. The values for this factor are given in the form of 1-D table.

The formula used to calculate the cumulative antenna ratio values for the metal layers is as follows:

© 2020 All Rights Reserved.

```
areaCar[j] += (((antennaMetalFactor*cumMetalArea) + (antennaCutFactor*cumCutArea))
     * antennaDiffAreaReduceFactor - diffArea * antennaDiffMinusFactor)/
     (cumGateAreaSharingCharge_[j] + antennaDiffPlusFactor * diffArea);
```

For more information, see the following:

- *Virtuoso Technology Data ASCII Files Reference*.
- Antenna calculation in Abstract Generator, see Antenna Tab.
- Process antenna effects, see Appendix D, “Calculating and Fixing Process Antenna Violations” in the *LEF/DEF 5.8 Language Reference*.

Example

```
constraintGroups(
 ( "foundry"
   antennaModels(
   ;( model
   ;( ----
    ( "default"
      antenna(
      ;( ratioName      layer     value )
      ;( ---------      -----     ----- )
       ( areaRatio      METAL1    0.345                  )
       ( diffAreaRatio  METAL1   ((0.45 0.4)(0.55 0.6))  )
       ( areaRatio      VIA1      0.344                  )
       ( diffAreaRatio  VIA1     ((0.44 0.5)(0.57 0.8))  )
      ) ;antenna

      cumulativeMetalAntenna(
      ;( ratioName         value )
      ;( ---------         ----- )
       ( areaRatio         0.544 'side  )
       ( diffAreaRatio     0.888        )
      ) ;cumulativeMetalAntenna

      cumulativeViaAntenna(
      ;( ratioName         value )
      ;( ---------         ----- )
       ( areaRatio         0.654 'side  )
       ( diffAreaRatio     0.884        )
      ) ;cumulativeViaAntenna
    ) ;default

    ( "second"
      antenna(
      ;( ratioName      layer     value )
      ;( ---------      -----     ----- )
       ( areaRatio      METAL1    1.345 'side               )
       ( diffAreaRatio  METAL1   ((0.455 0.4)(0.555 0.65))  )
      ) ;antenna


antenna(
( areaRatio "poly" <value> )
); antenna
```


© 2020 All Rights Reserved.

```
    cumulativeMetalAntenna(
    ;( ratioName         value )
    ;( ---------         ----- )
     ( areaRatio         3.564        )
     ( diffAreaRatio     7.888        )
    ) ;cumulativeMetalAntenna

    cumulativeViaAntenna(
    ;( ratioName         value )
    ;( ---------         ----- )
     ( areaRatio         7.654        )
     ( diffAreaRatio     1.884        )
    ) ;cumulativeViaAntenna
  ) ;second

  ( "third"
    antenna(
    ;( ratioName      layer     value )
    ;( ---------      -----     ----- )
     ( areaRatio      METAL1    0.343                     )
     ( diffAreaRatio  METAL1   ((0.43 0.3)(0.53 0.6))     )
    ) ;antenna

    cumulativeMetalAntenna(
    ;( ratioName         value )
    ;( ---------         ----- )
     ( areaRatio         9.765 'side  )
     ( diffAreaRatio     0.188        )
    ) ;cumulativeMetalAntenna

    cumulativeViaAntenna(
    ;( ratioName         value )
    ;( ---------         ----- )
     ( areaRatio         3.22                                )
     ( diffAreaRatio    ((0.421 0.4)(0.7 0.64)(0.81 0.59))    )
    ) ;cumulativeViaAntenna
  ) ;third

  ( "fourth"
    antenna(
    ;( ratioName      layer     value )
    ;( ---------      -----     ----- )
     ( areaRatio      METAL2    0.645 'side               )
     ( diffAreaRatio  METAL2   ((0.47 0.4)(0.55 0.7))     )
    ) ;antenna

    cumulativeMetalAntenna(
    ;( ratioName         value )
    ;( ---------         ----- )
     ( areaRatio         0.6                          )
     ( diffAreaRatio     ((0.127 0.23)(0.32 0.7))     )
    ) ;cumulativeMetalAntenna

    cumulativeViaAntenna(
    ;( ratioName         value )
    ;( ---------         ----- )
     ( areaRatio         9.67     )
     ( diffAreaRatio     0.684    )
    ) ;cumulativeViaAntenna
  ) ;fourth
```


© 2020 All Rights Reserved.

```
      ( "factors"
      ( cumulativeRoutingPlusCut "METAL1" t )
      ( area "METAL1" 0.50 )
      ( diffPlus "METAL1" 0.60 )
      ( diffMinus "METAL1" 0.70 )
      ( diffAreaReduce "METAL1" (
      ( 0.0 20.0 )
      ( 0.5 30.0 )
      ( 1.0 40.0 )
    ))
    ) ;factors
  ) ;antennaModels

   ) ;antennaModels
 ) ;foundry
) ;constraintGroups
```

Specific Requirements for Metal Density Calculation

All layers being abstracted should have a minimum density value specified in the technology file. The maximum density value is al. If a step size exists for both `maxDensity` and `minDensity` constraints, the value should match.

```
constraintGroups(
 ("foundry"
   spacings(
     ("minDensity" tx_layer g_density)
       ...
   ) ;spacings
  ) ;foundry
);constraintGroups

constraintGroups(
 ("foundry"
   spacingTables(
      ("minDensity" tx_layer
      ( ("step" nil nil) [ g_defaultValue ] )
      (g_table) )
   ) ;spacingTables
  ) ;foundry
);constraintGroups
```

Specific Technology Requirements for Viewing Data In Layout

The section describes the technology file requirements for viewing your data using Virtuoso tools.

Display packets are defined in the display resource file and control how layers appear on your monitor and plotting devices. You assign a display packet to a particular layer-purpose pair and specify other layer display attributes in the `techDisplays` subsection of the `layerDefinitions` section.

© 2020 All Rights Reserved.

For more information, see the following:

- *Virtuoso Technology Data User Guide*.
- *Virtuoso Technology Data ASCII Files Reference*.

Specific Requirements for Specifying Place-and-Route Information

On CDB, the `prRules` class defined the rules for place-and-route applications.

The `prRules` class does not exist in the technology file on OpenAccess. Information stored in the `prRules` class of the CDB technology file is stored in different locations in the OpenAccess 2.2 technology file. Many of these PR rules are mapped to the `LEFDefaultRouteSpec` constraint group (or the specified constraint group) in the OpenAccess 2.2 technology file.

The table below summarizes the mapping between CDB and OpenAccess 2.2 for the sections required by the abstract generator:

CDB Technology File	OpenAccess Technology File
`prRoutingLayers` (Specifies the routing direction of layers used for routing)	A routing layer in the OpenAccess 2.2 technology file must be included in the `validLayers` constraint of the `LEFDefaultRouteSpec` constraint group (or the specified constraint group) *and* have an appropriate `functions` subsection (with function `metal` or `poly`) specified in the `layerRules` section. See LEFDefaultRouteSpec on page 650. See Layer Functions on page 651.
`prMastersliceLayers` (Specifies the masterslice layers in order)	An OpenAccess masterslice layer must be included in the `functions` subsection of the `layerRules` section with the function `poly`. See Layer Functions on page 651.

LEFDefaultRouteSpec

In OpenAccess 2.2, `routeSpecs` are mapped to constraint groups. One of these constraint groups is `LEFDefaultRouteSpec` (or the specified constraint group), which is supported by the abstract generator.

© 2020 All Rights Reserved.

Example

```
constraintGroups(
 ( "LEFDefaultRouteSpec"
    ;layer constraints
    interconnect(
    ( validLayers   ( METAL1 METAL2 METAL3 )     )
    ( validVias     ( M1_M2 M2_M1 )              )
    ( errorLayers   GATEOXIDE1)
    ) ;interconnect

    ; physical constraints
    spacings(
      ( minWidth                "METAL1"0.6      )
      ( minSpacing              "METAL1"0.4'soft )
      ( minDiagonalWidth        "METAL1"0.3      )
      ( minDiagonalSpacing      "METAL1"0.2      )
      ( minWireExtension        "METAL1" 0.1      )
    ) ;spacings

    orderedSpacings(
      ( minOppExtension        "METAL1"  "VIA1"(0.1 0.2))
    ) ;orderedSpacings

    routingGrids(
     ( horizontalPitch   M1    0.19  )
     ( horizontalOffset  M1    0.095 )
     ( verticalPitch     M1    0.19  )
     ( verticalOffset    M1    0.095 )
    ) ;routingGrids
  ) ;LEFDefaultRouteSpec
) ;constraintGroups
```

For more information about constraint groups, see *Virtuoso Technology Data Constraint Reference*.

Note: The `prMastersliceLayers`, `prRoutingLayers`, `prRules`, and `routeSpecs` constructs have been deprecated and will be removed in an upcoming release. Cadence recommends that you remove them from your technology files.

Layer Functions

Layer functions are defined in the `functions` subsection of the `layerRules` section and are required in OpenAccess to identify the types of layers in the process. If this information is

© 2020 All Rights Reserved.

missing for a particular layer, that layer will be missing from any LEF exported by the abstract generator.

Subsection	Description	Example

`functions`	Specifies the layer functions (materials).
	
	Valid function values include `poly`, `diff`, `cut`, `metal`, `ndiff`, `pdiff`, `nplus`, `pplus`, `nwell`, `pwell`, `li`, `unknown`, `recognition`.
	
	Note: `diff` is a new masterslice layer type in OpenAccess 2.2.
		`layerRules(`
		`  functions(`
		
		`  ;( layer  function  [maskNumber])`
		`  ;( -----  --------  ------------)`
		
		`  ( POLY  "poly"  1  )`
		`  ( DIFF  "diff"  2  )`
		`  ( CONT  "cut"  3  )`
		`  ( M1  "metal"  4  )`
		`  ( VIA12  "cut"  5  )`
		`  ( NDIFF  "ndiff"  6  )`
		`  ( DIFF2  "pdiff"  7  )`
		`  ( NPLUS  "nplus"  8  )`
		`  ( PPLUS  "pplus"  9  )`
		`  ( NWELL  "nwell"  10  )`
		`  ( PWELL  "pwell"  11  )`
		`  ( LI  "li"  12  )`
		
		`  );functions`
		`);layerRules`

For more information, see *Virtuoso Technology Data ASCII Files Reference*.

Potential Issues Related to the Technology File in the Virtuoso Design Environment

This section highlights some of the problems that you might encounter when using a Virtuoso design environment technology file in Abstract Generator.

- Setting DATABASE MICRONS in the Exported LEF
- Including USEMINSPACING Properties in the Exported LEF
- Fewer than Two Metal Layers Defined
- Layer Not Fully Defined
- Layer with Spacing Rules is a Derived Layer
- Via Layers Not Fully Defined

© 2020 All Rights Reserved.

- Upper Layer is Not a Routing Layer
- Antenna Values Must Not Equal Zero
- Minimum Width Value is Zero
- Minimum Width is Not a Multiple of the Manufacturing Grid
- Minimum Spacing Value Equals Zero
- Cut Layer Must Have Upper and Lower Routing Layer
- Specifying Routing Grids

Working with Read-Only Technology File

By default, Abstract Generator creates boundary, pin, and net layer purpose pairs for the layers for which it does not find boundary, pin, and net purposes in the technology file, and then tries to save these layer purpose pairs in the technology file. Because the technology file is read-only, the layer purpose pairs are not saved and a warning message is displayed.

To enable Abstract Generator to work with the read-only technology file, the `ReadOnlyTechnology` option has been added to Abstract Generator. When this option is set to `TRUE`, Abstract Generator does not try to save to the technology file the layer purpose pairs that it creates.

You can set the `ReadOnlyTechnology` option to TRUE as follows:

```
absSetOption("ReadOnlyTechnology" "true")
```

Setting DATABASE MICRONS in the Exported LEF

You can use one of the following methods to set the correct `DATABASE MICRONS` values in the LEF file exported from Abstract Generator:

- Import a technology LEF file to create a technology file

  If the imported technology file contains the following statement:

  ```
  UNITS
      DATABASE MICRONS 1000  ;
  END UNITS
  ```

  it will be used by Abstract Generator and will be output to LEF created by Abstract Generator. In OpenAccess, `DATABASE MICRONS` is output to the technology LEF only.

- Manually edit the value in the technology file in Virtuoso design environment

© 2020 All Rights Reserved.

In OpenAccess, the `DATABASE MICRONS` value is taken from the database units per user unit (`DBUPerUU)` value defined in the technology file. A sample of the OpenAccess technology file is shown below.

On CDB, the value for `DBUPerUU` for each cellview type can be stored in multiple locations in the technology file, such as the `prop.xx` file, attached technology file, and `.cdsenv` file.

On OpenAccess 2.2, the database units per user unit (**`DBUPerUU`**) is stored at a single location in the technology file — `viewTypeUnits` subsection of the `controls` section.

Example

```
controls(
 viewTypeUnits(
 ;( viewType            userUnit       dbuperuu           )
 ;( --------            --------       --------           )

  ( maskLayout          "micron"        2000            )
  ( schematic           "inch"          160             )
  ( schematicSymbol     "inch"          160             )
  ( netlist             "inch"          160             )
 ) ;viewTypeUnits
) ;controls
```

After changing the value of `dbuperuu`, recompile and save the technology file by using the Technology File Manager. If you want to use this updated technology file in an open Abstract Generator session, you will need to re-open the library to force the application to re-read the updated technology file (*File – Library – Open*).

Including USEMINSPACING Properties in the Exported LEF

Consider a Virtuoso design environment technology file created from a technology LEF file that contains the following:

```
USEMINSPACING OBS ON  ;
```

Abstract Generator will use this property as described below and will also export it in the abstract LEF.

Setting `USEMINSPACINGOBS` on a library affects the distance at which blockages are cut away from pins in the abstract produced by Abstract Generator.

- If `USEMINSPACINGOBS` is set to `TRUE` on the technology library, Abstract Generator will use the appropriate minimum spacing table in the technology file for obstruction geometries.
- If `USEMINSPACINGOBS` = `FALSE`, which is the default state, Abstract Generator will use the largest `minSpacing` value in the table for that layer.

© 2020 All Rights Reserved.

Note: Abstract Generator uses only the pin width value from the spacing table in the technology file. It does not use the length value.

In OpenAccess, this property appears in the `controls` section of the technology file.

Fewer than Two Metal Layers Defined

To use Abstract Generator, you must have two or more metal routing layers defined in the technology file. Define routing layers in the `validLayers` constraint in the `interconnect` constraint category of the `LEFDefaultRouteSpec` constraint group (or the specified constraint group). In addition, specify an appropriate `functions` type (for example, `metal`) in the `layerRules` section.

Related error message: ABS-10015.

Layer Not Fully Defined

Abstract Generator requires that all the layers used in the process are fully defined in the appropriate technology file sections. Abstract Generator issues the message ABS-217 if a layer is defined in the `LEFDefaultRouteSpec` (or the specified constraint group) but is not included in the `functions` subsection in the `layerRules` section or if the layer function in the `functions` subsection is not specified as `metal`.

Related error message: ABS-10016.

Layer with Spacing Rules is a Derived Layer

If a layer has spacing rules that are defined in the `spacingTables` section of a constraint group, Abstract Generator proceeds depending on the layer definition in other sections of the technology file.

- If the layer is also defined in the `functions` subsection of the `layerRules` section, it represents an ideal situation and Abstract Generator proceeds without issuing any messages.
- If the layer is not additionally defined in the `functions` subsection of the `layerRules` section, Abstract Generator stops further execution and issues the error message ABS-10021.
- If the layer is also defined in the `techDerivedLayers` section that is used for defining defined layers and is not defined in the `functions` subsection of the `layerRules` section, Abstract Generator issues the info message ABS-10059 and proceeds with

© 2020 All Rights Reserved.

further execution. You should evaluate this info message carefully because Abstract Generator does not support derived layers. If such a layer is encountered, Abstract Generator will completely ignore the layer; existing shapes on this layer will not be processed and no shapes will be added on this layer.

Via Layers Not Fully Defined

Abstract Generator issues an error if there are no via layers defined in the technology file.

Via layer information must be defined in `standardViaDefs` in the `viaDefs` section of the technology file. Abstract Generator requires fully-specified interlayer connectivity for all masterslice, cut, and routing layers used in the design.

Note: When you are specifying up and down layers for a specific via (cut) layer, the lower layer must be specified first, then the via layer, and then the upper layer.

Example

```
viaDefs(
 standardViaDefs(
 ;(viaDefName layer1 layer2 (cutLayer cutWidth  ;cutHeight [resistancePerCut])
 ;(cutRows cutCol (cutSpace))
 ;(layer1Enc) (layer2Enc) (layer1Offset)
 ;(layer2Offset) (origOffset)
 ;[implant1 (implant1Enc) [implant2
 ;(implant2Enc)]])
 ;(-------------------------------------------------------------------------)

  ( VIA12  METAL1  METAL2  ("VIA1" 0.2 0.2)
   (1 1 (0.0 0.0))
   (0.1 0.1) (0.2 0.2) (0.0 0.0) (0.0 0.0)
   (0.0 0.0)
  )
  ( VIA23  METAL2  METAL3  ("VIA2" 0.1 0.2)
   (1 1 (0.0 0.0))
   (0.1 0.1) (0.2 0.2) (0.0 0.0) (0.0 0.0)
   (0.0 0.0)
  )
  ( VIA23Array  METAL2  METAL3  ("VIA2" 0.2 0.2)
   (1 1 (0.3 0.3))
   (0.1 0.1) (0.1 0.1) (0.0 0.0) (0.0 0.0)
   (0.0 0.0)
  )
 ) ;standardViaDefs
) ;viaDefs
```

Related error message: ABS-20007.

Additionally, each via layer must also be defined in the `functions` subsection of the `layerRules` section and must have the function `cut`.

Related error message: ABS-10018.

© 2020 All Rights Reserved.

Upper Layer is Not a Routing Layer

Abstract Generator issues an error if the upper layer specified in the `standardViaDefs` section is not a routing layer. You must also specify the layers from lowest to highest, otherwise the technology check might indicate that you have the layers incorrectly ordered.

Related error message: ABS-10047.

Antenna Values Must Not Equal Zero

Abstract Generator checks all antenna values to ensure that their value is not `0`.

Minimum Width Value is Zero

Abstract Generator checks the `minWidth` values specified for all the metal layers in the design and issues an error message if the value is undefined or is `0`. The `minWidth` constraint is defined in the `spacings` constraint category of the technology file; For example,

```
constraintGroups(
 ( "foundry"
   ;physical constraints
   spacings(

   ;( constraint layer1 layer2 value )
   ;( ---------- ------ ------ ----- )

    ( minWidth   "metal1"      0.5 )
    ( minWidth   "metal2"      0.5 )
   ) ;spacings
 ) ;foundry
) ;constraintGroups
```

Related error message: ABS-10023.

Minimum Width is Not a Multiple of the Manufacturing Grid

Abstract Generator issues an error message if the `minWidth` value is not a multiple of the manufacturing grid for a particular layer. This check is performed only if there is a manufacturing grid value defined in the technology file.

Set the manufacturing grid value in the `controls` section and the minimum width value in the `spacings` constraint category of the technology file; For example,

```
controls(
  mfgGridResolution(
  ( 0.005000 )
  ) ;mfgGridResolution
```


© 2020 All Rights Reserved.

```
) ;controls

constraintGroups(
 ( "foundry"
   ;physical constraints
   spacings(
   ;( constraint layer1 layer2 value )
   ;( ---------- ------ ------ ----- )
    ( minWidth   "metal1"      0.5 )
    ( minSpacing "metal1"      0.5 )
    ( minWidth   "metal2"      0.5 )
    ( minSpacing "metal2"      0.5 )
   ) ;spacings
 ) ;foundry
) ;constraintGroups
```

Related error message: ABS-10022.

Minimum Spacing Value Equals Zero

The minimum spacing rule defines the minimum spacing required between objects on a single layer and is defined in either the `spacings` or `spacingTables` constraint category of the `foundry` constraint group in the technology file.

Abstract Generator checks the `minSpacing` value for every metal layer and issues an error message if the value is undefined or is set to `0`.

For more information about spacing rules, see Physical Rules.

Related error message: ABS-10024.

Note: A manufacturing grid check is also performed here.

Cut Layer Must Have Upper and Lower Routing Layer

Abstract Generator checks to ensure that every cut layer used in a default via has both an upper and a lower routing layer. It stops if this check fails.

Define vias in the `LEFDefaultRouteSpec` (`validVias`) and in a custom via definition (`customViaDefs`) in the `viaDefs` class.

Related error message: ABS-10030.

© 2020 All Rights Reserved.

Specifying Routing Grids

To enable Abstract to start, it is required to define routing grids in the technology file. If the routing grids are not defined in the technology file, the technology file will not load properly and an error message will be displayed.

Converting Common Lisp Files to SKILL or Tcl Format

This section describes how to convert any Abstract Generator replay files you have from common-lisp format into SKILL or Tcl format.

Because SKILL is used extensively in the Virtuoso design environment, Cadence recommends that you convert any replay files from common-lisp to SKILL format. To do this, use the Abstract Common Lisp Converter, `absclconv`.

Caution

You should use `absclconv` ***to convert only Abstract Generator common-lisp files. It is not designed as a generic common-lisp conversion utility.***

The command accepts the following arguments.

```
absclconv
     { -tcl | -skill }
     clFile
     newFile
```

These arguments are described in detail below.

`-tcl	-skill`	Specifies the format of the file to be generated from the common-lisp file.
`clFile`	Specifies the common-lisp file to be converted.
`newFile`	Specifies the name of the file to be created in the new format.

For ease of use, the following shortcuts are provided.

- To convert a common-lisp file directly to SKILL format, use the `abscl2il` link.

  `abscl2il` *`clFile ilFile`*

  This creates a new SKILL file, *`ilFile`*, based on the common-lisp file *`clFile`*.

- To convert a common-lisp file directly to Tcl format, use the `abscl2tcl` link.

© 2020 All Rights Reserved.

`abscl2tcl` *`clFile tclFile`*

This creates a new Tcl file, *`tclFile`*, based on the common-lisp file *`clFile`*.

In all cases, if a file with the same name already exists, it is overwritten.

`absclconv` preserves comments and blank lines, so a command that is commented out in the common-lisp version remains commented-out in the SKILL and Tcl versions. (Note, however, that comments are displayed differently in SKILL and Tcl.)

The example below shows how a common-lisp command converts to Tcl and SKILL formats.

Common-Lisp	Tcl	SKILL
`(abs:Select "param1" "param2")`	`absSelect "param1" "param2"`	`absSelect ("param1" "param2")`

© 2020 All Rights Reserved.

Index

Symbols

.abstract.options File 77, 306
.Xdefaults File 58
Customize Appearance of 60
Buttons 61
Entry Boxes 64
GUI 66
Labels 65
Tables 61
Text Entries 63
Customize Fonts 58

A

Abstract Generation
Abort Operation 57
Flows Supported 32
CDB Flow 34
DEF to OA 32
GDSII to LEF 32
LEF to OA 33
Prepare for 90
Use Model 29
Abstract Generator
Environment Setup 38
Launch
In Standalone Mode 39
Performance Optimization 85
Support for Inherited Connections 368
Abstract Step 377
Adjust Tab 166
Blockage Tab 190
Density Tab 207
Fracture Tab 212
Overlap Tab 220
Site Tab 216
Adjust Tab 166
Create Boundary Pins 167
Create Follow Ring Pins 171
Create Power Geometry Group 172
Create Ring Pins 170
Create Signal Nets 171
Layer Assignment 159
Antenna
Oxide 153
Antenna Calculation 142
Checks Performed 150
Keywords 150
Antenna Tab 139
Layer Assignment 151, 155
Relationship with Signal Tab 126
Types of Antenna Calculations 142
Automatic Pin Stretching 204

B

Bin Selector 346
Bins
Add 341
Delete 343
Options 344
Rename 342
Work with 49
Blockage Model 192
Cover 192
Detailed 192
Shrink 192
Blockage Tab 190
Layer Assignment 191
Blocks Tab 105
Create Power Pins 105
Preserve Local Blockages 105
Boundary Pins
Non-Square 168
Square 167
Boundary Tab 102
Adjust Boundary 104
Create Boundary 102
Fix Boundary 104

C

Cell Abstracts 75
Cell Reports 54
Cells
Deselect 355
Distribute 75, 359
Edit 372

© 2020 All Rights Reserved.

Move 362
Properties 363
Report 54, 371
Revalidate 356
Select 352, 354
Sort 357
Terminal Properties 366
CLASS CORE Ports
Cell Edge Facing Core 175
Copy Ports to LEF 177
Create Multiple Ports 176
Include Pins 177
Pins Meet Cell Boundary 177
Pins Meet Non-Core Facing Edge 178
Include Power/Ground Nets 175
Create Blockage
Cut Below 196
Cut on Same Layer 195
Distance From Boundary 197
Shrink Distance 196
Shrink Track 196
Create Pins
Boundary Pins 167
Follow Ring Pins 171
Must Connect Pins
Power Nets 138
Must Join Pins 132
Power Pins 105
Rings Pins 170
Customize
GUI 58
Cut Out Around Pins
On Layer Below 196
On Same Layer 195
Size 202

D

Density Tab 207

E

Error Messages 565
Exclude Metal Purpose Blockage 123
Export
LEF 324
Options 328
Report 327
Extract
Power Nets 136
Signal Nets 128
Extract Step 113, 376
Antenna Tab 139
General Tab 157
Power Tab 135
Signal Tab 125
Extraction Limits 122
Depth 122
Distance 122
Width 123

F

File
General Options 332
Record 329
Replay 331
Fonts
Apply 67
Fracture Tab 212
45-Degree Geometry 213
Fracture Blockages 213
Fracture Pins 212

G

General Tab 157
Graphical User Interface 37
Cell Pane 51
Command Entry Field 55
Command History Window 56
Form Features 302
Log Window 54
Main Window 43
Menu Bar 44
Monitor Progress 57
Toolbar 46
Ground Sensitivity 368

H

Help Commands 379
Hooks 68
Sample 71

© 2020 All Rights Reserved.

I

Ignore Purposes 333
Import
  Options 323
Import Data 74
  DEF 318
  LEF 315
  Logical 321
  Stream (GDSII) 309, 312
Inherited Connections
  Ground Sensitivity 368
  Supply Sensitivity 368

L

Layer Assignment Table for
  Antenna Extraction 155
  Antenna Regions 151
  Blockages 191
  Pin Geometry 159
  Power Extraction 136
  Signal Extraction 129
Layer Map File 310, 313
Library
  Attach Technology Information 308
  Create 304
  Open 306

M

Map Tab 91
  Map Text Labels to Pins 92
Menus
  Bin 45
  Cells 45
  File 45
  Flow 45
  Help 46
Migration Issues
  Converting Common Lisp Files 659

N

Net Shapes 182
  Abutment 182
  Feedthru 182
newlink
    ExtractAntennaExcludeCumPerLayerAntennaRatio 469

O

Overlap Tab 220
  Create Overlap Boundary 221
  Overlap Boundary
    Size Factor 221
    Smooth Factor 221

P

Pins Step 89, 375
  Blocks Tab 105
  Boundary Tab 102
  Map Tab 91
  Text Tab 98
Power and Ground Rail
  Characteristics 181
  Net 182
  Offset 183
  Scenarios 183
  Shape 182
  Width 183
Power Tab 135
  Extract Power Nets 136
  Layer Assignment 136

R

Regular Expressions
  Examples 562
  Special Characters 560
  Special Constructs 562
  Syntax Overview 559
  Write 95

S

Signal Tab 125
  Extract Signal Nets 128
  Layer Assignment 129
  Relationship with Antenna Tab 126
Site Tab 216
  Calculate Site Pattern 219

© 2020 All Rights Reserved.

Define Site Name 218
Select Site Name 218
Stair-Step Coverage 214
Full 214
Partial 215
Truncate 214
Stair-Steps Width 215
Step Selector 346
Supply Sensitivity 368

T

Technology File
Overwrite Existing 316
Text Tab 98
Manipulate Text Labels 99
Preserve Text Labels 100

V

Views
Status Information 52

© 2020 All Rights Reserved.