

LEF/DEF Language Reference

Product Version 5.4
January 2003

© 1990-2003 Cadence Design Systems, Inc. All rights reserved.
Printed in the United States of America.

Cadence Design Systems, Inc., 555 River Oaks Parkway, San Jose, CA 95134, USA

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. (Cadence) 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 Print Permission: This publication is protected by copyright and any unauthorized use of this publication may violate copyright, trademark, and other laws. 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. This statement grants you permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used solely for personal, informational, and noncommercial purposes;
2. The publication may not be modified in any way;
3. Any copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement; and
4. Cadence reserves the right to revoke this authorization at any time, and any such use shall be discontinued immediately upon written notice from Cadence.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. The information contained herein is the proprietary and confidential information of Cadence or its licensors, and is supplied subject to, and may be used only by Cadence's customer in accordance with, a written agreement between Cadence and its customer. 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

<u>Preface</u>	9
<u>What's New</u>	9
<u>Related Documents</u>	9
<u>Typographic and Syntax Conventions</u>	10
<u>Character Information</u>	11
 <u>1</u>	
<u>LEF Syntax</u>	13
<u>About Library Exchange Format Files</u>	14
<u>General Rules</u>	14
<u>Order of LEF Statements</u>	14
<u>LEF Statement Definitions</u>	15
<u>Antenna Size</u>	15
<u>Bus Bit Characters</u>	16
<u>Clearance Measure</u>	16
<u>Divider Character</u>	16
<u>Extensions</u>	17
<u>Layer (Cut)</u>	17
<u>Layer (Masterslice or Overlap)</u>	21
<u>Layer (Routing)</u>	23
<u>Macro</u>	38
<u>Manufacturing Grid</u>	55
<u>Names Case Sensitive</u>	56
<u>Nondefault Rule</u>	56
<u>No Wire Extension</u>	60
<u>Property Definitions</u>	60
<u>Same-Net Spacing</u>	62
<u>Site</u>	63
<u>Units</u>	64
<u>Use Min Spacing</u>	66
<u>Version</u>	67

LEF/DEF 5.4 Language Reference

<u>Via</u>	67
<u>Via Rule</u>	71
<u>Via Rule Generate</u>	74

2

DEFINE and ALIAS

<u>DEFINE Statements</u>	79
<u>DEFINE Expressions</u>	80
<u>DEFINE Syntax</u>	83
<u>DEFINE Expansion</u>	84
<u>DEFINE Examples</u>	85
<u>ALIAS Statements</u>	86
<u>ALIAS Definition</u>	86
<u>ALIAS Examples</u>	86
<u>ALIAS Expansion</u>	87

3

Working with LEF

<u>Incremental LEF</u>	89
<u>Error Checking</u>	90
<u>Message Facility</u>	91
<u>Error-Checking Facility</u>	93

4

DEF Syntax

<u>About Design Exchange Format Files</u>	96
<u>General Rules</u>	96
<u>Order of DEF Statements</u>	97
<u>DEF Statement Definitions</u>	98
<u>Blockages</u>	98
<u>Bus Bit Characters</u>	99
<u>Components</u>	100
<u>Design</u>	102
<u>Die Area</u>	103

LEF/DEF 5.4 Language Reference

<u>Divider Character</u>	103
<u>Extensions</u>	103
<u>Fills</u>	104
<u>GCell Grid</u>	105
<u>Groups</u>	106
<u>History</u>	108
<u>Names Case Sensitive</u>	108
<u>Nets</u>	108
<u>Pins</u>	117
<u>Pin Properties</u>	124
<u>Property Definitions</u>	125
<u>Regions</u>	128
<u>Rows</u>	129
<u>Scan Chains</u>	130
<u>Slots</u>	134
<u>Special Nets</u>	135
<u>Technology</u>	142
<u>Tracks</u>	142
<u>Units</u>	143
<u>Version</u>	144
<u>Vias</u>	145

A

<u>Examples</u>	147
<u>LEF</u>	147
<u>DEF</u>	157
<u>Scan Chain Synthesis Example</u>	162

B

<u>Via Pattern Encoding Convention</u>	165
<u>Overview</u>	165
<u>Bitmapped Via Pattern Encoding Convention</u>	165
<u>Name Form</u>	166
<u>Cut Representation</u>	166
<u>Compressed Via Pattern Encoding Convention</u>	166

LEF/DEF 5.4 Language Reference

<u>Character Sets</u>	167
<u>Rules for Double-Decimal Digit Character Set</u>	167

C

<u>Optimizing LEF Technology for Place and Route</u>	171
<u>Overview</u>	171
<u>Guidelines for No Wire Extension at Pin</u>	172
<u>Guidelines for Routing Pitch</u>	172
<u>Guidelines for Wide Metal Spacing</u>	175
<u>Guidelines for Wire Extension at Vias</u>	176
<u>Guidelines for Default Vias</u>	178
<u>Guidelines for Stack Vias (MAR Vias) and Samenet Spacing</u>	180
<u>Example of an Optimized LEF Technology File</u>	184

D

<u>Calculating and Fixing Process Antenna Violations</u>	191
<u>Overview</u>	192
<u>What Are Process Antennas?</u>	193
<u>What Is the Process Antenna Effect (PAE)?</u>	194
<u>What Is the Antenna Ratio?</u>	195
<u>What Can Be Done to Improve the Antenna Ratio?</u>	195
<u>Using Process Antenna Keywords in the LEF and DEF Files</u>	196
<u>Calculating Antenna Ratios</u>	197
<u>Calculating the Antenna Area</u>	197
<u>Calculating a PAR</u>	198
<u>Calculating a CAR</u>	202
<u>Calculating Ratios for a Cut Layer</u>	210
<u>Checking for Antenna Violations</u>	212
<u>Area Ratio Check</u>	213
<u>Side Area Ratio Check</u>	213
<u>Cumulative Area Ratio Check</u>	214
<u>Cumulative Side Area Ratio Check</u>	215
<u>Example Using the Antenna Keywords</u>	216
<u>Using Antenna Diode Cells</u>	217
<u>Changing the Routing</u>	217

LEF/DEF 5.4 Language Reference

<u>Inserting Antenna Diode Cells</u>	218
<u>Using DiffUseOnly</u>	218
<u>Calculations for Hierarchical Designs</u>	219
<u>LEF and DEF Keywords for Hierarchical Designs</u>	220
<u>Design Example</u>	220
<u>Top-Down Hierarchical Design Example</u>	223
 <u>Index</u>	 225

LEF/DEF 5.4 Language Reference

Preface

This manual is a language reference for users of the Cadence® Library Exchange Format (LEF) and Design Exchange Format (DEF) integrated circuit (IC) description languages.

LEF defines the elements of an IC process technology and associated library of cell models. DEF defines the elements of an IC design relevant to physical layout, including the netlist and design constraints. LEF and DEF inputs are in ASCII form.

This manual assumes that you are familiar with the development and design of integrated circuits.

This preface provides the following information:

- [What's New](#) on page 9
- [Related Documents](#) on page 9
- [Typographic and Syntax Conventions](#) on page 10
- [Character Information](#) on page 11

What's New

For information on what is new or changed in LEF and DEF for version 5.4, see [What's New in LEF/DEF](#).

Related Documents

The following Cadence tools read LEF and DEF files and write layout data in DEF. For information about these tools, see the related documents in the Cadence online documentation system (CDSDoc).

- Silicon Ensemble® Place-and-Route
- Cadence Ultra Placer
- Cadence Ultra Router
- Cadence Clock Tree Generator

LEF/DEF 5.4 Language Reference

Preface

■ Pearl® Timing Analyzer

Note: References in this manual to specific place-and-route tools are meant to note how those tools must use the LEF or DEF syntax. They do not refer to using LEF or DEF syntax in general. Other place-and-route tools might also have limitations when using LEF and DEF syntax. For information on any limitations, see the documentation for the specific tool.

Typographic and Syntax Conventions

This list describes the conventions used in this manual.

<code>text</code>	Words in <code>monospace</code> type indicate keywords that you must enter literally. These keywords represent language tokens.
<i>variable</i>	Words in <i>italics</i> indicate user-defined information for which you must substitute a name or a value.
<i>objRegExpr</i>	An object name with the identifier <i>objRegExpr</i> represents a regular expression for the object name.
<i>pt</i>	Represents a point in the design. This value corresponds to a coordinate pair, such as <code>x y</code> . You must enclose a point within parentheses, with space between the parentheses and the coordinates. For example, <code>RECT (1000 2000) (1500 400) .</code>
	Vertical bars separate possible choices for a single argument. They take precedence over any other character.
[]	Brackets denote optional arguments. When used with vertical bars, they enclose a list of choices from which you can choose one.
{ }	Braces used with vertical bars enclose a list of choices from which you must choose one.
...	Three dots indicate that you can repeat the previous argument. If they are used with brackets, you can specify zero or more arguments. If they are used with braces, you must specify at least one argument, but you can specify more.

LEF/DEF 5.4 Language Reference

Preface

, . . .	A comma and three dots together indicate that if you specify more than one argument, you must separate those arguments with commas.
" "	Quotation marks enclose string values. Write quotation marks within a string as \". Write a backslash within a string as \\\.

Any characters not included in the list above are required by the language and must be entered literally.

Character Information

LEF and DEF 5.4 support the following characters:

!	< and >
\$. (period)
&	?
	{ }
:	' (single quotation mark)
`	" (double quotation mark)
@	_ (underbar)
~	^
=	, (comma)
Uppercase and lowercase alphabet characters	Numbers

DEF 5.4 reserves the following characters for special functions:

()	Coordinates
+	Start of new keyword
-	Coordinates and start of new keyword
[]	Default special characters for bus bits unless overridden by <code>BUSBITCHARS</code>
/	Default special character for hierarchy unless overridden by <code>DIVIDERCHAR</code>

LEF/DEF 5.4 Language Reference

Preface

LEF and DEF names cannot contain the following ASCII characters:

<code>\n</code>	Newline
	Space
<code>;</code>	Semicolon

Note: LEF and DEF names also cannot contain the ASCII character used by the place-and-route tool for comments. For example, if the tool uses the pound sign (#) for comments, it cannot be used in a LEF or DEF name.

LEF and DEF interpret the following characters as regular expressions. Use these characters in LEF and DEF names only when you intend a regular expression.

<code>*</code>	Asterisk	Matches any sequence of characters
<code>%</code>	Percent	Matches any single character

Note: Pattern matching only works in a few areas of the DEF file, such as component name matching in the `SPECIALNET` section, and component name matching in `GROUP` definitions.

You can use the backslash (`\`) as an escape character. When the backslash precedes a character that has special meaning in LEF or DEF, the special meaning of the character is ignored. The following characters have special meanings that can be escaped:

<code>BUSBITCHARS</code> <i>delimiterPair</i>	The characters you specify to enclose bus bits
<code>DIVIDERCHAR</code> <i>character</i>	The character you specify to express hierarchy
<code>\</code>	The backslash character

Note: You cannot use the escape character with the pound sign (#).

LEF Syntax

This chapter contains information about the following topics:

- [About Library Exchange Format Files](#) on page 14
 - ❑ [General Rules](#) on page 14
 - ❑ [Order of LEF Statements](#) on page 14
- [LEF Statement Definitions](#) on page 15
 - ❑ [Antenna Size](#) on page 15
 - ❑ [Bus Bit Characters](#) on page 16
 - ❑ [Clearance Measure](#) on page 16
 - ❑ [Divider Character](#) on page 16
 - ❑ [Extensions](#) on page 17
 - ❑ [Layer \(Cut\)](#) on page 17
 - ❑ [Layer \(Masterslice or Overlap\)](#) on page 21
 - ❑ [Layer \(Routing\)](#) on page 23
 - ❑ [Macro](#) on page 38
 - [Layer Geometries](#) on page 44
 - [Macro Obstruction Statement](#) on page 46
 - [Macro Pin Statement](#) on page 49
 - ❑ [Manufacturing Grid](#) on page 55
 - ❑ [Names Case Sensitive](#) on page 56
 - ❑ [Nondefault Rule](#) on page 56
 - ❑ [No Wire Extension](#) on page 60

LEF/DEF 5.4 Language Reference

LEF Syntax

- ❑ [Property Definitions](#) on page 60
- ❑ [Same-Net Spacing](#) on page 62
- ❑ [Site](#) on page 63
- ❑ [Units](#) on page 64
- ❑ [Use Min Spacing](#) on page 66
- ❑ [Version](#) on page 67
- ❑ [Via](#) on page 67
- ❑ [Via Rule](#) on page 71
- ❑ [Via Rule Generate](#) on page 74

About Library Exchange Format Files

A Library Exchange Format (LEF) file contains library information for a class of designs. Library data includes layer, via, placement site type, and macro cell definitions. The LEF file is an ASCII representation using the syntax conventions described in [“Typographic and Syntax Conventions”](#) on page 10.

General Rules

Note the following information about creating LEF files:

- Lines in the LEF file are limited to 2,048 characters (extra characters are truncated).
- Distance is specified in microns.
- Distance precision is controlled by the `UNITS` statement.
- LEF statements end with a semicolon (`;`). You must leave a space between the last character in the statement and the semicolon.

Order of LEF Statements

LEF files can contain the following statements. You must specify the statements in the following order:

```
VERSION statement
NAMECASESENSITIVE statement
[ NOWIREEXTENSIONATPIN statement ]
```

LEF/DEF 5.4 Language Reference

LEF Syntax

```
BUSBITCHARS statement
DIVIDERCHAR statement
[ UNITS statement ]
[ MANUFACTURINGGRID statement ]
[ USEMINSPACING statement ]
[ CLEARANCEMEASURE statement ]
[ PROPERTYDEFINITIONS statement ]
[ ANTENNASIZE statement ]
{ LAYER (Nonrouting) statement
  | LAYER (Routing) statement } ...
VIA statement
{ VIARULE statement
  | VIARULE GENERATE statement } ...
[ NONDEFAULTRULE statement ]
[ SPACING statement ]
{ SITE statement } ...
{ MACRO statement
  [ PIN statement ] ...
  [ OBS statement ... ] } ...
[ BEGINEXT statement ] ...
END LIBRARY
```

Note: The header section (VERSION, NAMESCASESENSITIVE, BUSBITCHARS and DIVIDERCHAR) is required. However, the NOWIREEXTENSIONATPIN statement is library dependent, and you may not need it.

LEF Statement Definitions

The following definitions describe the syntax arguments for the statements that make up a LEF file. Statements are listed in alphabetical order, *not* in the order they must appear in a LEF file. For the correct order, see [“Order of LEF Statements”](#) on page 14.

Antenna Size

```
[ ANTENNAINPUTGATEAREA value ;]
[ ANTENNAINPUTDIFFAREA value ;]
[ ANTENNAOUTPUTDIFFAREA value ;]
```

Defines global default antenna area values.

Note: Global default area values also apply to any external I/O pins in the DEF file (ANTENNAOUTPUTDIFFAREA applies to DEF input pins, ANTENNAINPUTGATEAREA applies to DEF output pins, and ANTENNAINPUTDIFFAREA applies to DEF inout pins). However, a global value is often too optimistic for DEF output pins because it implies a diffusion connection through *metal1*, which is normally incorrect for DEF output pins. If you use the

LEF/DEF 5.4 Language Reference

LEF Syntax

global area values, Cadence recommends you specify an `ANTENNADIFFAREA` value for all DEF output pins.

`ANTENNAINOUTDIFFAREA` *value*

Specifies the default inout pin diffusion area, in microns squared.

`ANTENNAINPUTGATEAREA` *value*

Specifies the default input pin gate area, in microns squared.

`ANTENNAOUTPUTDIFFAREA` *value*

Specifies the default output pin diffusion area, in microns squared.

Bus Bit Characters

`BUSBITCHARS` "*delimiterPair*" ;

Specifies the pair of characters used to specify bus bits when LEF names are mapped to or from other databases. The characters must be enclosed in double quotation marks. For example:

`BUSBITCHARS` "()" ;

If one of the bus bit characters appears in a LEF name as a regular character, you must use a backslash (\) before the character to prevent the LEF reader from interpreting the character as a bus bit delimiter.

Clearance Measure

[`CLEARANCEMEASURE` {`MAXXY` | `EUCLIDEAN`} ;]

Defines the clearance spacing requirement that will be applied to pins and obstructions (blockages) in cells. If you do not specify a `CLEARANCEMEASURE` statement, euclidean distance is used by default.

`MAXXY` Uses the largest x or y distances for spacing between objects.

`EUCLIDEAN` Uses the euclidean distance for spacing between objects. That is, the square root of $x^2 + y^2$.

Divider Character

`DIVIDERCHAR` "*character*" ;

LEF/DEF 5.4 Language Reference

LEF Syntax

Specifies the character used to express hierarchy when LEF names are mapped to or from other databases. The character must be enclosed in double quotation marks. For example:

```
DIVIDERCHAR "/" ;
```

If the divider character appears in a LEF name as a regular character, you must use a backslash (\) before the character to prevent the LEF reader from interpreting the character as a hierarchy delimiter.

Extensions

```
[BEGINEXT "tag"
    extension
ENDEXT]
```

Adds customized syntax to the LEF file that can be ignored by tools that do not use that syntax. You can also use extensions to add new syntax not yet supported by your version of LEF/DEF, if you are using version 5.1 or later.

extension Specifies the contents of the extension.

"*tag*" Identifies the extension block. You must enclose *tag* in quotes.

Example

```
BEGINEXT "lVSI Signature 1.0"
    CREATOR "company name"
    DATE "timestamp"
    REVISION "revision number"
ENDEXT
```

Layer (Cut)

```
LAYER layerName
    TYPE CUT ;
    [SPACING minSpacing [LAYER 2ndLayerName] ;] ...
    [PROPERTY propName propVal ;] ...
    [ACCURRENTDENSITY {PEAK | AVERAGE | RMS}
        { value
            | FREQUENCY freq_1 freq_2 ... ;
            [CUTAREA cutArea_1 cutArea_2 ... ;]
        TABLEENTRIES
            v_freq_1_cutArea_1 v_freq_1_cutArea_2 ...
            v_freq_2_cutArea_1 v_freq_2_cutArea_2 ...
            ...
        ]
    ]
```

LEF/DEF 5.4 Language Reference

LEF Syntax

```
    } ;]
[DCCURRENTDENSITY AVERAGE
  { value
  | CUTAREA cutArea_1 cutArea_2 ... ;
    TABLEENTRIES value_1 value_2 ...
  } ;]
[ANTENNAAREARATIO value ;]
[ANTENNADIFFAREARATIO {value | PWL ( ( d1 r1 ) ( d2 r2 ) ... )} ;]
[ANTENNACUMAREARATIO value ;]
[ANTENNACUMDIFFAREARATIO {value | PWL ( ( d1 r1 ) ( d2 r2 ) ... )} ;]
[ANTENNAAREAFACOR value ;]

END layerName
```

Defines cut layers in the design. Each cut layer is defined by assigning it a name and design rules. You must define cut layers separately, with their own layer statements.

You must define layers in process order from bottom to top. If multiple masterslice layers are defined, the cut layer corresponding to a masterslice layer must immediately follow the last masterslice layer. For example:

```
well      masterslice
diffusion masterslice
poly      masterslice
cut01     cut
metal1    routing
cut12     cut
metal2    routing
cut23     cut
metal3    routing
```

ACCURRENTDENSITY Specifies how much AC current a cut of a certain area can handle at a certain frequency. The ACCURRENTDENSITY syntax is defined as follows:

```
{PEAK | AVERAGE | RMS}
{ value
| FREQUENCY freq_1 freq_2 ... ;
  [CUTAREA cutArea_1 cutArea_2 ... ;]
  TABLEENTRIES
    v_freq_1_cutArea_1 v_freq_1_cutArea_2 ...
    v_freq_2_cutArea_1 v_freq_2_cutArea_2 ...
    ...
} ;:
```

PEAK Specifies the peak limit value of the wire.

AVERAGE Specifies the average limit value of the wire.

LEF/DEF 5.4 Language Reference

LEF Syntax

RMS	Specifies the root mean square limit value of the wire.
<i>value</i>	Specifies a value for the wire. <i>Type:</i> Float
FREQUENCY	Specifies frequency values, in megahertz. You can specify more than one frequency value. <i>Type:</i> Float
CUTAREA	Specifies cut area values, in square microns. You can specify more than one cut area value. <i>Type:</i> Float
TABLEENTRIES	Defines the current density for each of the frequency and via cut area pairs specified in the FREQUENCY and CUTAREA statements, in milliamperes. The pairing defines each cut area for the first frequency in the FREQUENCY statement, then the cut areas for the second frequency, and so on. <i>Type:</i> Float

ANTENNAAREAFACITOR *value*

Specifies the multiply factor for the antenna metal area calculation. For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

Default: 1.0

Type: Float

Note: If you specify a value that is greater than 1.0, the computed areas will be larger, and violations will occur more frequently.

ANTENNAAREARATIO *value*

Specifies the maximum legal antenna ratio, using the area of the metal wire that is not connected to the diffusion diode. For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

Type: Integer

LEF/DEF 5.4 Language Reference

LEF Syntax

ANTENNACUMAREARATIO *value*

Specifies the cumulative antenna ratio, using the area of the metal wire that is not connected to the diffusion diode. For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

Type: Integer

ANTENNACUMDIFFAREARATIO {*value* | PWL ((*d1 r1*) (*d2 r2*) ...) }

Specifies the cumulative antenna ratio, using the area of the metal wire that is connected to the diffusion diode. You can supply an explicit ratio *value* or specify piece-wise linear format (PWL), in which case the cumulative ratio is calculated using linear interpolation of the diffusion area and ratio input values. The diffusion input values must be specified in ascending order. For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

Type: Integer

ANTENNADIFFAREARATIO {*value* | PWL ((*d1 r1*) (*d2 r2*) ...) }

Specifies the antenna ratio, using the area of the metal wire connected to the diffusion diode. You can supply an explicit ratio *value* or specify piece-wise linear format (PWL), in which case the ratio is calculated using linear interpolation of the diffusion area and ratio input values. The diffusion input values must be specified in ascending order. For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

Type: Integer

DCCURRENTDENSITY AVERAGE

Specifies how much DC current a cut of a certain area can handle at a certain frequency. The DCCURRENTDENSITY syntax is defined as follows:

```
AVERAGE
{ value
| CUTAREA cutArea_1 cutArea_2 ... ;
  TABLEENTRIES value_1 value_2 ...
};
```

AVERAGE Specifies the average limit value for the wire.

value Specifies a value for the wire.

Type: Float

LEF/DEF 5.4 Language Reference

LEF Syntax

CUTAREA	Specifies cut area values, in square microns. You can specify more than one cut area value. <i>Type:</i> Float
TABLEENTRIES	Specifies the value of current density for each specified cut area, in milliamperes. <i>Type:</i> Float
LAYER <i>LayerName</i>	Specifies the name for the layer. This name is used in later references to the layer.
PROPERTY <i>propName propVal</i>	Specifies a numerical or string value for a layer property defined in the PROPERTYDEFINITIONS statement. The <i>propName</i> you specify must match the <i>propName</i> listed in the PROPERTYDEFINITIONS statement.
SPACING <i>minSpacing</i> [LAYER <i>2ndLayerName</i>]	Specifies the minimum spacing allowed between via cuts on the same net or different nets. This value can be overridden by the SAMENET SPACING statement. You can also make the spacing value apply between objects on the cut layer being defined and objects on <i>2ndLayerName</i> . The second cut layer must already be defined in the LEF file.
TYPE CUT	Specifies that the layer is for contact-cuts. The type is later referenced in vias and in rules for generating vias.

Layer (Masterslice or Overlap)

```
LAYER layerName
    TYPE {MASTERSLICE | OVERLAP} ;
    [PROPERTY propName propVal ;] ...
END layerName
```

Defines masterslice (nonrouting) or overlap layers in the design. Each layer is defined by assigning it a name and design rules. You must define masterslice or overlap layers separately, with their own layer statements.

Note: Currently, most libraries do not use masterslice layers because all of their are on a routing layer.

LEF/DEF 5.4 Language Reference

LEF Syntax

You must define layers in process order from bottom to top. If multiple masterslice layers are defined, the cut layer corresponding to a masterslice layer must immediately follow the last masterslice layer. For example:

```
well      masterslice
diffusion masterslice
poly      masterslice
cut01     cut
metall     routing
cut12     cut
metal2     routing
cut23     cut
metal3     routing
```

LAYER *layerName* Specifies the name for the layer. This name is used in later references to the layer.

TYPE Specifies the purpose of the layer.

MASTERSLICE Layer is fixed in the base array. If pins appear in the masterslice layers, you must define vias to permit the routers to connect those pins and the first routing layer. Wires are not allowed on masterslice layers.

Routing tools can use only one masterslice layer. If a masterslice layer is defined, exactly one cut layer must be defined between the masterslice layer and the adjacent routing layers.

OVERLAP Layer used for overlap checking for rectilinear blocks. Obstruction descriptions in the macro obstruction statements refer to the overlap layer.

PROPERTY *propName propVal* Specifies a numerical or string value for a layer property defined in the **PROPERTYDEFINITIONS** statement. The *propName* you specify must match the *propName* listed in the **PROPERTYDEFINITIONS** statement.

LEF/DEF 5.4 Language Reference

LEF Syntax

Layer (Routing)

```
LAYER layerName
    TYPE ROUTING ;
    DIRECTION {HORIZONTAL | VERTICAL} ;
    PITCH distance ;
    WIDTH defWidth ;
    [OFFSET distance ;]
    [AREA minArea ; ]
    [SPACING minSpacing
        [ RANGE minWidth maxWidth
            [ USELENGTHTHRESHOLD
                | INFLUENCE value [RANGE stubMinWidth stubMaxWidth]
                | RANGE minWidth maxWidth]
            | LENGTHTHRESHOLD maxLength [RANGE minWidth maxWidth] ]
        ;] ...
    [WIREEXTENSION value ; ]
    [MINIMUMCUT numCuts WIDTH minWidth ;]
    [RESISTANCE RPERSQ value ;]
    [CAPACITANCE CPERSQDIST value ;]
    [HEIGHT distance ;]
    [THICKNESS distance ;]
    [SHRINKAGE distance ;]
    [CAPMULTIPLIER value ;]
    [EDGECAPACITANCE value ;]
    [SLOTWIREWIDTH minWidth ;]
    [SLOTWIRELENGTH minLength ;]
    [SLOTWIDTH minWidth ;]
    [SLOTLENGTH minLength ;]
    [MAXADJACENTSLOTSPACING spacing ;]
    [MAXCOAXIALSLOTSPACING spacing ;]
    [MAXEDGESLOTSPACING spacing ;]
    [SPLITWIREWIDTH minWidth ;]
    [MINIMUMDENSITY minDensity ;]
    [MAXIMUMDENSITY maxDensity ;]
    [DENSITYCHECKWINDOW windowLength windowWidth ;]
    [DENSITYCHECKSTEP stepValue ;]
    [FILLACTIVESPACING spacing ;]
    [ANTENNAAREARATIO value ;]
    [ANTENNADIFFAREARATIO {value | PWL ( ( d1 r1 ) ( d2 r2 ) ... ) } ;]
    [ANTENNACUMAREARATIO value ;]
    [ANTENNACUMDIFFAREARATIO {value | PWL ( ( d1 r1 ) ( d2 r2 ) ... ) } ;]
    [ANTENNAAREAFACOR value [DIFFUSEONLY] ;]
    [ANTENNASIDEAREARATIO value ;]
    [ANTENNADIFFSIDEAREARATIO {value | PWL ( ( d1 r1 ) ( d2 r2 ) ... ) } ;]
    [ANTENNACUMSIDEAREARATIO value ;]
    [ANTENNACUMDIFFSIDEAREARATIO {value | PWL ( ( d1 r1 ) ( d2 r2 ) ... ) } ;]
    [ANTENNASIDEAREAFACOR value [DIFFUSEONLY] ;]
    [PROPERTY propName propVal ;] ...
    [ACCURRENTDENSITY {PEAK | AVERAGE | RMS}
        { value
```

LEF/DEF 5.4 Language Reference

LEF Syntax

```
| FREQUENCY freq_1 freq_2 ... ;  
  [WIDTH width_1 width_2 ... ;]  
  TABLEENTRIES  
    v_freq_1_width_1 v_freq_1_width_2 ...  
    v_freq_2_width_1 v_freq_2_width_2 ...  
    ...  
  } ;]  
[DCCURRENTDENSITY AVERAGE  
  { value  
  | WIDTH width_1 width_2 ... ;  
    TABLEENTRIES value_1 value_2 ...  
  } ;]
```

END *layerName*

Defines routing layers in the design. Each layer is defined by assigning it a name and design rules. You must define routing layers separately, with their own layer statements.

You must define layers in process order from bottom to top. For example:

```
well      masterslice  
diffusion masterslice  
poly      masterslice  
cut01     cut  
metal1    routing  
cut12     cut  
metal2    routing  
cut23     cut  
metal3    routing
```

ACCURRENTDENSITY Specifies how much AC current a wire of a certain width can handle at a certain frequency. The ACCURRENTDENSITY syntax is defined as follows:

```
{PEAK | AVERAGE | RMS}  
{ value  
  | FREQUENCY freq_1 freq_2 ... ;  
    [WIDTH width_1 width_2 ... ;]  
    TABLEENTRIES  
      v_freq_1_width_1 v_freq_1_width_2 ...  
      v_freq_2_width_1 v_freq_2_width_2 ...  
      ...  
  } ;
```

PEAK Specifies the peak limit value of the wire.

AVERAGE Specifies the average limit value of the wire.

LEF/DEF 5.4 Language Reference

LEF Syntax

RMS	Specifies the root mean square limit value of the wire.
<i>value</i>	Specifies a value for the wire. <i>Type:</i> Float
FREQUENCY	Specifies frequency values, in megahertz. You can specify more than one frequency. <i>Type:</i> Float
WIDTH	Specifies wire width values, in microns. You can specify more than one wire width. <i>Type:</i> Float
TABLEENTRIES	Defines the current density for each of the frequency and width pairs specified in the FREQUENCY and WIDTH statements, in milliamperes. The pairing defines each width for the first frequency in the FREQUENCY statement, then the widths for the second frequency, and so on. <i>Type:</i> Float

ANTENNAAREAFACOR *value* [DIFFUSEONLY]

Specifies the multiply factor for the antenna metal area calculation. DIFFUSEONLY specifies that the current antenna factor should only be used when the corresponding layer is connected to the diffusion.

Default: 1.0

Type: Float

For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

Note: If you specify a value that is greater than 1.0, the computed areas will be larger, and violations will occur more frequently.

ANTENNAAREARATIO *value*

Specifies the maximum legal antenna ratio, using the area of the metal wire that is not connected to the diffusion diode. For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

Type: Integer

LEF/DEF 5.4 Language Reference

LEF Syntax

ANTENNACUMAREARATIO *value*

Specifies the cumulative antenna ratio, using the area of the wire that is not connected to the diffusion diode. For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

Type: Integer

ANTENNACUMDIFFAREARATIO {*value* | PWL ((*d1 r1*) (*d2 r2*) ...) }

Specifies the cumulative antenna ratio, using the area of the metal wire that is connected to the diffusion diode. You can supply and explicit ratio *value* or specify piece-wise linear format (PWL), in which case the cumulative ratio value is calculated using linear interpolation of the diffusion area and ratio input values. The diffusion input values must be specified in ascending order.

Type: Integer

For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

ANTENNACUMDIFFSIDEAREARATIO {*value* | PWL ((*d1 r1*) (*d2 r2*) ...) }

Specifies the cumulative antenna ratio, using the side wall area of the metal wire that is connected to the diffusion diode. You can supply and explicit ratio *value* or specify piece-wise linear format (PWL), in which case the cumulative ratio value is calculated using linear interpolation of the diffusion area and ratio input values. The diffusion input values must be specified in ascending order.

Type: Integer

For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

ANTENNACUMSIDEAREARATIO *value*

Specifies the cumulative antenna ratio, using the side wall area of the metal wire that is not connected to the diffusion diode. For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

ANTENNADIFFAREARATIO {*value* | PWL ((*d1 r1*) (*d2 r2*) ...) }

Specifies the antenna ratio, using the area of the metal wire that is connected to the diffusion diode. You can supply and explicit ratio *value* or specify piece-wise linear format (PWL), in which

LEF/DEF 5.4 Language Reference

LEF Syntax

case the ratio value is calculated using linear interpolation of the diffusion area and ratio input values. The diffusion input values must be specified in ascending order.

Type: Integer

For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

ANTENNADIFFSIDEAREARATIO { *value* | PWL ((*d1 r1*) (*d2 r2*) ...) }

Specifies the antenna ratio, using the side wall area of the metal wire that is connected to the diffusion diode. You can supply an explicit ratio *value* or specify piece-wise linear format (PWL), in which case the ratio value is calculated using linear interpolation of the diffusion area and ratio input values. The diffusion input values must be specified in ascending order.

Type: Integer

For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

ANTENNASIDEAREAFACOR *value*

Specifies the multiply factor for the antenna metal side wall area calculation. DIFFUSEONLY specifies that the current antenna factor should only be used when the corresponding layer is connected to the diffusion.

Default: 1.0

Type: Float

For more information on process antenna calculation, see ["Calculating and Fixing Process Antenna Violations"](#) in the *Ultra Router Reference*.

ANTENNASIDEAREARATIO *value* [DIFFUSEONLY]

Specifies the antenna ratio, using the side wall area of the metal wire that is not connected to the diffusion diode. For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

Type: Integer

AREA *minArea*

Specifies the minimum metal area required for polygons on the layer, in distance units squared. *minArea* is the smallest allowed area of a single shape on a layer.

Type: Float

LEF/DEF 5.4 Language Reference

LEF Syntax

CAPACITANCE CPERSQDIST *value*

Specifies the capacitance for each square unit, in picofarads per square micron. This is used to model wire-to-ground capacitance.

CAPMULTIPLIER *value*

Specifies the multiplier for interconnect capacitance to account for increases in capacitance caused by nearby wires.

Default: 1

Type: Integer

DCCURRENTDENSITY

Specifies how much DC current a wire of a certain width can handle at a certain frequency. The DCCURRENTDENSITY syntax is defined as follows:

```
AVERAGE
{ value
| WIDTH width_1 width_2 ... ;
  TABLEENTRIES value_1 value_2 ...
} ;
```

AVERAGE Specifies the average limit value of the wire.

value Specifies a value for the wire.

WIDTH Specifies wire width values, in microns. You can specify more than one wire width.
Type: Float

TABLEENTRIES Specifies the value of current density for each specified width, in milliamperes.
Type: Float

DENSITYCHECKSTEP *stepValue*

Specifies the stepping distance for metal density checks, in distance units.

Type: Float

DENSITYCHECKWINDOW *windowLength windowWidth*

Specifies the dimensions of the check window, in distance units.

Type: Float

DIRECTION {HORIZONTAL | VERTICAL}

Specifies the preferred routing direction. The automatic routing tools attempt to route in the preferred direction on a layer. A

LEF/DEF 5.4 Language Reference

LEF Syntax

typical case is to route horizontally on layers 1 and 3, and vertically on layer 2.

EDGECAPACITANCE *value*

Specifies a floating-point value of peripheral capacitance, in picofarads per micron. The place-and-route tool uses this value in two situations:

- Estimating capacitance before routing
- Calculating segment capacitance after routing

For the second calculation, the tool uses *value* only if you set layer thickness, or layer height, to 0. In this situation, the peripheral capacitance is used in the following formula:

segment capacitance = (layer capacitance per square x segment width x segment length) + (peripheral capacitance x 2 (segment width + segment length)).

For Silicon Ensemble, if you do not specify a value for EDGECAPACITANCE, the values of the Timing.Fringcap.layernum environmental variables are used instead.

FILLACTIVESPACING *spacing*

Specifies the spacing between metal fills and active geometries.
Type: Float

HEIGHT *distance*

Specifies the distance from the top of the ground plane to the bottom of the interconnect.
Type: Float

LAYER *layerName*

Specifies the name for the layer. This name is used in later references to the layer.

MAXADJACENTSLOTSPACING *spacing*

Specifies the maximum spacing, in distance units, allowed between two adjacent slot sections.
Type: Float

MAXCOAXIALSLOTSPACING *spacing*

Specifies the maximum spacing, in distance units, allowed

LEF/DEF 5.4 Language Reference

LEF Syntax

between two slots in the same slot section.

Type: Float

MAXEDGESLOTSPACING *spacing*

Specifies the maximum spacing, in distance units, allowed between slot edges.

Type: Float

MAXIMUMDENSITY *maxDensity*

Specifies the maximum metal density percentage allowed for the layer. The *minDensity* and *maxDensity* values represent the metal density range within which all areas of the design must fall. The metal density must be greater than or equal to *minDensity* and less than or equal to *maxDensity*.

Type: Float

For example, the following syntax specifies a metal density range in which the minimum metal density must be greater than or equal to 20 percent and the maximum metal density must be less than or equal to 70 percent.

```
MINIMUMDENSITY 20.0 ;
```

```
MAXIMUMDENSITY 70.0 ;
```

MINIMUMCUT *numCuts* WIDTH *minWidth*

Specifies the number of cuts a via must have when it is on a wide wire or pin whose width is greater than *minWidth*.

MINIMUMDENSITY *minDensity*

Specifies the minimum metal density percentage allowed for the layer. The *minDensity* and *maxDensity* values represent the metal density range within which all areas of the design must fall. The metal density must be greater than or equal to *minDensity* and less than or equal to *maxDensity*.

Type: Float

OFFSET *distance*

Specifies the offset, in distance units, from the placement grid to the routing grid for the layer.

Default: Half the routing pitch for the layer

Type: Float

PITCH *distance*

Specifies the routing pitch for the layer. Pitch is used to generate the routing grid. For more information, see [“Routing Pitch”](#) on

LEF/DEF 5.4 Language Reference

LEF Syntax

page 34.

Type: Float

PROPERTY *propName propVal*

Specifies a numerical or string value for a layer property defined in the PROPERTYDEFINITIONS statement. The *propName* you specify must match the *propName* listed in the PROPERTYDEFINITIONS statement.

RESISTANCE RPERSQ *value*

Specifies the resistance for a square of wire, in ohms per square. The resistance of a wire can be defined as
 $RPERSQU \times \text{wire length/wire width}$

SHRINKAGE *distance*

Specifies the value to account for shrinkage of interconnect wiring due to the etching process. Actual wire widths are determined by subtracting this constant value.

Type: Float

SLOTLENGTH *minLength*

Specifies the minimum slot length, in distance units, allowed in the design.

Default: The minimum spacing for the layer

Type: Float

SLOTWIDTH *minWidth*

Specifies the minimum slot width, in distance units, allowed in the design.

Default: The minimum spacing for the layer

Type: Float

SLOTWIRELENGTH *minLength*

Specifies the minimum wire length, in distance units, allowed for wires that need to be slotted.

Type: Float

SLOTWIREWIDTH *minWidth*

Specifies the minimum wire width, in distance units, allowed for wires that need to be slotted.

Type: Float

SPACING

Specifies the spacing rules to use for wiring on the layer. For information on and examples of using spacing rules, see [“Using Spacing Rules”](#) on page 34.

LEF/DEF 5.4 Language Reference

LEF Syntax

The SPACING syntax is defined as follows:

```
[SPACING minSpacing
  [ RANGE minWidth maxWidth
    [ USELENGTHTHRESHOLD
      | INFLUENCE value
        [RANGE stubMinWidth stubMaxWidth]
      | RANGE minWidth maxWidth]
    | LENGTHTHRESHOLD maxLength
      [RANGE minWidth maxWidth] ]
;] ...
SPACING minSpacing
  [RANGE minWidth maxWidth]
```

Specifies the default minimum spacing, in microns, allowed between two geometries on different nets.

Type: Float

The minimum spacing rule applies to objects on the layer with widths in the indicated RANGE (that is, widths that are greater than or equal to *minWidth* and less than or equal to *maxWidth*). If you do not specify a range, the rule applies to all objects.

Type: Float

USELENGTHTHRESHOLD

Specifies that the threshold spacing rule should be used if the other object meets the previous LENGTHTHRESHOLD value.

```
INFLUENCE value
  [RANGE stubMinWidth stubMaxWidth]
```

Specifies the area of the stub wire that inherits the spacing from a wider wire.

Type: Float

The influence rule applies to stub wires on the layer with widths in the indicated RANGE (that is, widths that are greater than or equal to *stubMinWidth* and less than or equal to *stubMaxWidth*). If you do not specify a range, the rule applies to all stub wires.

Type: Float

LEF/DEF 5.4 Language Reference

LEF Syntax

`RANGE minWidth maxWidth`

Specifies an optional second width range. The spacing rule applies if the widths of both objects fall in the ranges defined (each object in a different range). For an object's width to fall in a range, it must be greater than or equal to *minWidth* and less than or equal to *maxWidth*.

Type: Float

`LENGTHTHRESHOLD maxLength`
`[RANGE minWidth maxWidth]`

Specifies the maximum parallel run length or projected length with an adjacent metal object.

The threshold spacing rule applies to objects with widths in the indicated `RANGE` (that is, widths that are greater than or equal to *minWidth* and less than or equal to *maxWidth*). If you do not specify a range, the rule applies to all objects.

Type: Float

`SPLITWIREWIDTH minWidth`

Specifies the minimum wire width, in distance units, allowed for wires that need to be split.

Type: Float

`THICKNESS distance`

Specifies the thickness of the interconnect.

Type: Float

`TYPE ROUTING`

Identifies the layer as a routable layer.

`WIDTH defWidth`

Specifies the default routing width to use for all regular wiring on the layer. This is the minimum metal width allowed on the layer.

Type: Float

`WIREEXTENSION value`

Specifies the distance by which wires are extended at vias. You must specify a value that is more than half of the routing width.

Default: Wires are extended half of the routing width

Type: Float

LEF/DEF 5.4 Language Reference

LEF Syntax

Routing Pitch

The `PITCH` statements define the detail routing grid generated when you initialize a floorplan. The pitch for a given routing layer defines the distance between tracks in the preferred direction for that layer. The complete routing grid is the union of the tracks generated for each routing layer and these tracks apply to all routing layers.

The spacing of the grid should be no less than line-to-via spacing in both the horizontal and vertical directions. Grid spacing less than line-to-via spacing can result in routing problems and in a decrease in the utilization results.

The grid should allow for diagonal vias. Via spacing on all layers included in the via definition in LEF determines whether or not diagonal vias can be used. The router is capable of avoiding violations between diagonal vias. If you allow diagonal vias, less time is needed for routing and the layout creates a smaller design.

Pitch Ratio in Three-Layer Designs

For three-layer designs, the ratio of the pitch of the metal1 layer to the pitch of the metal3 layer should be reasonable. For example, a 1:1 pitch ratio is ideal, and 1:2 or 1:3 is reasonable. But a ratio of 7:8 can make it very difficult for the router to place vias.

In general, make the metal1 pitch larger to get a 1:1 ratio of the metal1 pitch to the metal3 pitch. In a three-layer design, much of the metal1 routing space is blocked by the internal geometries of cells, and this ratio maximizes the routing space available in metal3.

Using Spacing Rules

Spacing rules apply to pin-to-wire, obstruction-to-wire, via-to-wire, and wire-to-wire spacing. These requirements specify the default minimum spacing allowed between two geometries on different nets.

When defined with a `RANGE` argument, a spacing value applies to all objects with widths within a specified range. That is, the rule applies to objects whose widths are greater than or equal to the specified minimum width and less than or equal to the specified maximum width.

In the following example, the default minimum allowed spacing between two adjacent objects is .3 microns. However, for objects between 1.01 and 2.0 microns in width, the spacing is 0.5 microns.

```
SPACING 0.3 ;
```

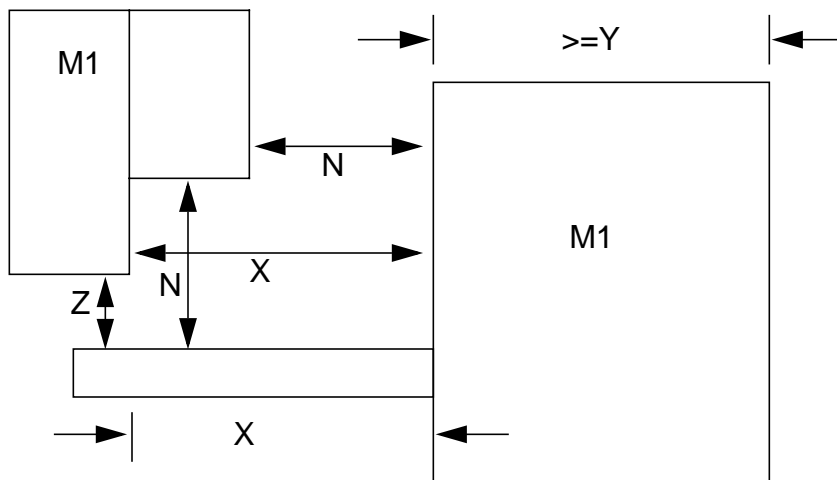
```
SPACING 0.5 RANGE 1.01 2.0 ;
```

LEF/DEF 5.4 Language Reference

LEF Syntax

Influence spacing rules are used to support the inheritance of wide wire spacing by nets connected to the wide wires. For example, a larger spacing is needed for stub wires attached to large objects like pre-routed power wires. A piece of metal connecting to a wider wire will inherit spacing rules for a user-defined distance from the wider wire.

In the following illustration, a minimum space of N is required between two metal lines when at least one metal line has a width of $\geq Y$. This spacing must be maintained for any small piece of metal ($< Y$) that is connected to the wide metal within X range of the wide metal. Outside of this range, normal spacing rules (Z) apply.



In the following example, the .5 micron spacing applies for the first 2.0 microns of the stub sticking out from the large object. This rule only applies to the stub wire; the previous rule must be included for the wide wire spacing.

```
SPACING 0.28 ;  
    SPACING 0.20 LENGTHTHRESHOLD 1.5 ;  
  
SPACING 0.5  
    RANGE 2.01 2000.0  
    INFLUENCE 2.000 ;
```

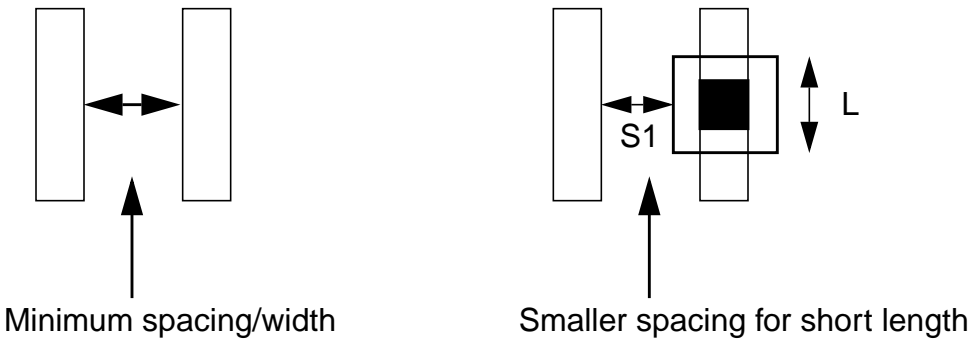
Some processes only need the INFLUENCE rule for certain widths of the stub wire. In the following example, the .5 micron spacing is required only for stub wires between .5 and 1.0 microns in width.

```
SPACING 0.5  
    RANGE 2.01 2000.0  
    INFLUENCE 2.00 RANGE 0.5 1.0 ;
```

LEF/DEF 5.4 Language Reference

LEF Syntax

Threshold spacing is a function of both the wire width and the length of the neighboring object. It is typically used when vias are wider than the wire to allow tighter wire-to-wire spacing, even when the vias are present.



In the following example, a slightly tighter spacing of .24 microns is needed if the other object is less than or equal to 1 micron in length.

```
SPACING 0.28 ;  
SPACING 0.24 LENGTHTHRESHOLD 1.0 ;
```

The **USELENGTHTHRESHOLD** argument specifies that the threshold spacing rule should be applied if the other object meets the previous **LENGTHTHRESHOLD** value.

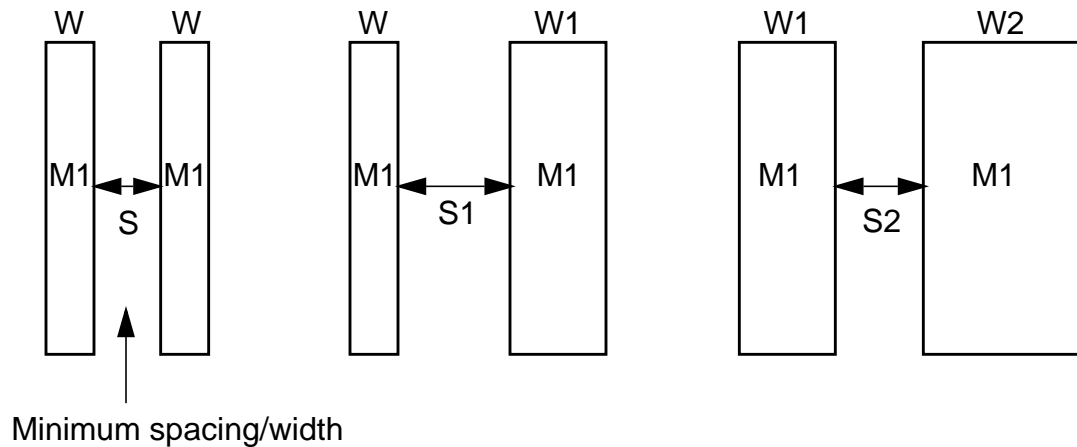
In the following example, a larger spacing of .32 microns is needed for wire widths between 1.01 and 9.99 microns. However, if the other object is less than or equal to 1 micron in length, the smaller .28 micron spacing is applied.

```
SPACING 0.32  
    RANGE 1.01 9.99 USELENGTHTHRESHOLD ;
```

LEF/DEF 5.4 Language Reference

LEF Syntax

For complex wide wire spacing, the spacing of the wires is a function of the width of both wires. You can use a second `RANGE` argument to meet the spacing requirements of wide wires.



In the following example, the spacing between two adjacent objects is 1 micron, when the width of one object is between 1.01 and 2.00 microns, and the width of the other object is between 2.01 and 3.00 microns.

```
SPACING 1.00 RANGE 1.01 2.00 RANGE 2.01 3.00 ;
```

Note: For different spacing requirements, there cannot be overlapping ranges in both the first and second ranges at the same time. However, it is possible to have overlapping spacings in one of the ranges, as long as the other range does not overlap.

Examples

The following examples define current density tables:

```
LAYER met1
...
ACCURRENTDENSITY PEAK                                #peak AC current limit for met1
FREQUENCY 1E6 100E6 400E6 ;                          #3 freq values in MHz
WIDTH 0.4 0.8 10.0 50.0 100.0 ;                     #5 width values in microns
TABLEENTRIES
2.0E-6 1.9E-6 1.8E-6 1.7E-6 1.5E-6                  #mAmps for 5 widths and freq_1
1.4E-6 1.3E-6 1.2E-6 1.1E-6 1.0E-6                  #mAmps for 5 widths and freq_2
0.9E-6 0.8E-6 0.7E-6 0.6E-6 0.4E-6 ;                #mAmps for 5 widths and freq_3

ACCURRENTDENSITY AVERAGE                             #avg. AC current limit for met1
FREQUENCY 1E6 100E6 400E6 ;                          #3 freq values in MHz
TABLEENTRIES 0.6E-6 0.5E-6 0.4E-6 ;                  #mAmps of 0.6 for 3 freq
```

LEF/DEF 5.4 Language Reference

LEF Syntax

```
ACCURRENTDENSITY RMS                #RMS AC current limit for met1
FREQUENCY 100E6 400E6 800E6 ;        #3 freq values in MHz
WIDTH      0.4 0.8 10.0 50.0 100.0 ; #5 width values in microns
TABLEENTRIES
2.0E-6 1.9E-6 1.8E-6 1.7E-6 1.5E-6  #mAmps for 5 widths and freq_1
1.4E-6 1.3E-6 1.2E-6 1.1E-6 1.0E-6  #mAmps for 5 widths and freq_2
0.9E-6 0.8E-6 0.7E-6 0.6E-6 0.4E-6 ;#mAmps for 5 widths and freq_3
...
END met1 ;

LAYER cut12
...
ACCURRENTDENSITY PEAK                #peak AC current limit for via cut12
FREQUENCY 1E6 100E6 ;                #2 freq values in MHz
TABLEENTRIES
0.5E-6 0.4E-6 ;                      #mAmps for each of the 2 freq values

ACCURRENTDENSITY AVERAGE            #average AC current limit for via cut12
FREQUENCY 1E6 100E6 ;                #2 freq values in MHz
TABLEENTRIES
0.6E-6 0.5E-6 ;                      #mAmps for each of the 2 freq values

ACCURRENTDENSITY RMS                #RMS AC current limit for via cut12
FREQUENCY 100E6 800E6;               #2 freq values in MHz
TABLEENTRIES
0.5E-6 0.4E-6 ;                      #mAmps for each of the 2 freq values
....
END cut12 ;
```

Macro

```
MACRO macroName
[CLASS
{ COVER
| RING
| BLOCK
| PAD [INPUT | OUTPUT | INOUT | POWER | SPACER]
| CORE [FEEDTHRU | TIEHIGH | TIELOW | SPACER | ANTENNACELL]
| ENDCAP {PRE | POST | TOPLEFT | TOPRIGHT | BOTTOMLEFT | BOTTOMRIGHT}
}
;]
[SOURCE {USER | BLOCK} ;]
[FOREIGN foreignCellName [pt [orient]] ;]
[ORIGIN pt ;]
[EEQ macroName ;]
[LEQ macroName ;]
[SIZE width BY height ;]
[SYMMETRY {X | Y | R90} ... ;]
[SITE siteName ;]
```

LEF/DEF 5.4 Language Reference

LEF Syntax

```
[PIN statement] ...  
[OBS statement] ...  
[PROPERTY propName propVal ;] ...
```

END *macroName*

Defines macros in the design.

CLASS	Specifies the macro type. If you do not specify CLASS, the macro is considered a CORE macro, and a warning prints when the LEF file is read in. You can specify macros of the following types:
COVER	Macro with data that is fixed to the floorplan and cannot change, such as power routing (ring pins) around the core.
RING	Large macro that can cut prerouted special nets and connect to these nets with ring pins.
BLOCK	Predefined macro used in hierarchical design
PAD	I/O pad. A pad can be one of the following types: INPUT, OUTPUT, INOUT, POWER, or SPACER, for I/O rows; INPUT, OUTPUT, INOUT, or POWER, for I/O corner pads.
CORE	Macro used in the core area. A core macro can be one of the following types: FEEDTHRU—Used for connecting to another cell. TIEHIGH,TIELOW—Used for connecting unused I/O terminals to the power or ground bus. SPACER—Sometimes called a filler cell, this cell is used to fill in space between regular core cells. ANTENNACELL—Used for solving process antenna violations. This cell has a single input to a diode to bleed off charge that builds up during manufacturing.

LEF/DEF 5.4 Language Reference

LEF Syntax

ENDCAP A macro placed at the ends of core rows (to connect with power wiring).

If the library includes only one corner I/O macro, then appropriate **SYMMETRY** must be included in its macro description. An **ENDCAP** macro can be one of the following types:

PRE—A left-end macro
POST—A right-end macro
TOPLEFT—A top left I/O corner cell
TOPRIGHT—A top right I/O corner cell
BOTTOMLEFT—A bottom left I/O corner cell
BOTTOMRIGHT—A bottom right I/O corner cell

EEQ *macroName*

Specifies that the macro being defined should be electrically equivalent to the previously defined *macroName*. For Silicon Ensemble, **PLACE** automatically selects among **EEQ** macros to obtain the best placement. **EEQ** macros include devices such as OR-gates or inverters that have several implementations with different shapes, geometries, and orientations.

Electrically equivalent macros have the following requirements:

- Corresponding pins must have corresponding functionality.
- Pins must be defined in the same order.
- For each group of corresponding pins (one from each macro), pin function and electrical characteristics must be the same.
- The **EEQ** *macroName* specified must refer to a previously defined macro. If the **EEQ** *macroName* referenced is already electrically equivalent to other model macros, all referenced macros are considered electrically equivalent.

FOREIGN *foreignCellName* [*pt* [*orient*]]

Specifies the foreign (GDSII) structure name to use when placing an instance of the macro. The optional *pt* coordinate specifies the macro origin (lower left corner when the macro is in north orientation) offset from the foreign origin. The optional *orient* value specifies the orientation of the foreign cell when the macro is in north orientation.

LEF/DEF 5.4 Language Reference

LEF Syntax

For Silicon Ensemble, the calculation of absolute location for the foreign reference in `OUTPUT GDSII` is affected by component placement location and component orientation, as well as by the macro foreign coordinate offset, but not by the macro foreign orientation.

`LEQ macroName`

Specifies that the macro being defined should be logically equivalent to the previously defined *macroName*. Unlike EEQ macros, LEQ macros can have pins in which the corresponding electrical characteristics, such as timing, resistance, and capacitance, are different.

LEQ macros can be replaced by one another without effecting an Engineering Change. Automatic layout commands do not replace a macro in the netlist with an LEQ macro.

An example of an LEQ macro is a distribution cell that has several implementations with different electrical characteristics, such as drive strengths, intrinsic delays, and so on.

The following are requirements of LEQ macros:

- As with EEQ macros, the number, specified order, and corresponding functions of the pins on LEQ macros must be the same.
- Internal nonschematic mustjoin and power pins can be different.
- Pin geometries, obstructions, size, foreign references, orientation, power/grid connections, and site requirements can be different.
- The LEQ *macroName* must have been previously defined. If the LEQ *macroName* referenced is already logically equivalent to other macros, all referenced macros are considered logically equivalent.

`MACRO macroName`

Specifies the name of the library macro.

`OBS statement`

Defines obstructions on the macro. Obstruction geometries are specified using layer geometries syntax. See [“Macro Obstruction Statement”](#) on page 46 for syntax information.

LEF/DEF 5.4 Language Reference

LEF Syntax

ORIGIN <i>pt</i>	Specifies the origin of the macro. <i>pt</i> is the lower left corner of the macro. The coordinates for macro sites, ports, and obstructions are specified with respect to the macro origin. The origin itself is specified with respect to the lower left corner of the bounding box of the sites of the macro.				
PIN <i>statement</i>	Defines pins for the macro. See “Macro Pin Statement” on page 49 for syntax information.				
PROPERTY <i>propName propVal</i>	Specifies a numerical or string value for a macro property defined in the PROPERTYDEFINITIONS statement. The <i>propName</i> you specify must match the <i>propName</i> listed in the PROPERTYDEFINITIONS statement.				
SITE <i>siteName</i>	Specifies the site associated with the macro. You can specify only one site. For Silicon Ensemble, this is the same as the row type specified in the ADD ROW command when floorplanning the design. The legal location checking for placement consists of mapping the macro site to the row sites to see if the name is the same and the orientations match.				
SIZE <i>width BY height</i>	Specifies the minimum bounding rectangle, in microns, for the macro. The bounding rectangle should be a multiple of the placement grid, to allow for abutting cells. The bounding rectangle normally contains all pin and blockage geometries.				
SOURCE {USER BLOCK}	<p>Specifies the source of the macro being defined.</p> <p><i>Value:</i> Specify one of the following:</p> <table><tr><td>USER</td><td>Macro is user-defined.</td></tr><tr><td>BLOCK</td><td>Block macro is created when the LEF file is written out.</td></tr></table>	USER	Macro is user-defined.	BLOCK	Block macro is created when the LEF file is written out.
USER	Macro is user-defined.				
BLOCK	Block macro is created when the LEF file is written out.				
SYMMETRY {X Y R90}	Specifies the allowable orientations for the macro. For example, SYMMETRY Y indicates that N and FN orientations are legal candidates. N is always a legal candidate. For each type of symmetry defined, additional orientations become legal candidates.				

LEF/DEF 5.4 Language Reference

LEF Syntax

For Silicon Ensemble, the specification of legal candidates helps the legal location checks in `PLACE` and `MOVE CELL`. If a candidate does not have legal positions on the floorplan, there is a slight loss of efficiency but not of functionality.

For corner I/O pads, if the library includes `BOTTOMLEFT`, `BOTTOMRIGHT`, `TOPLEFT`, and `TOPRIGHT` I/O corner cells, then they are placed in north orientation (no flipping). However, if the library includes only one type of corner I/O, then `SYMMETRY` in `x` and `y` are required to create the rows for all four of them.

Defining Cover Macros

If you define a cover macro with its actual size, some place-and-route tools cannot place the rest of the cells in your design because it uses the cell boundary to check for overlaps. You can resolve this in two ways:

- The easiest way to support a cover macro is to define the cover macro with a small size, for example, 1 by 1. Silicon Ensemble does not require port figures to be within the cell boundary.
- If you want to define the cover macro with its actual size, create an overlap layer with the `LAYER (nonrouting) TYPE OVERLAP` statement. You define this overlap layer (cover macro) with the macro obstruction (`OBS`) statement. This method slows down the `PLACE` command slightly.

Examples

The following examples show two variations of the `FOREIGN` argument. The negative offset specifies that the GDSII structure should be above and to the right of the macro lower left corner.

```
MACRO ABC ...  
FOREIGN ABC -2 -3 ;
```

The positive offset specifies that the GDSII structure should be below and to the left of the macro lower left corner.

```
MACRO EFG ...  
FOREIGN EFG 2 3 ;
```

LEF/DEF 5.4 Language Reference

LEF Syntax

Layer Geometries

```
{ LAYER layerName [SPACING minSpacing | DESIGNRULEWIDTH value] ;
  [WIDTH width ;]
  { PATH pt ... ;
    | PATH ITERATE pt ... stepPattern ;
    | RECT pt pt ;
    | RECT ITERATE pt pt stepPattern ;
    | POLYGON pt pt pt pt ... ;
    | POLYGON ITERATE pt pt pt pt ... stepPattern ;
  } ...
  | VIA pt viaName ;
  | VIA ITERATE pt viaName stepPattern ;
}
```

Used in the macro obstruction (OBS) and pin port (PIN) statements to define layer geometries in the design.

DESIGNRULEWIDTH *value*

Specifies the effective design rule width. If specified, the router uses the spacing defined in the layer section corresponding to a width of *value*. This overrides the normal LAYER-based spacing rules for this particular obstruction or pin. If you specify the DESIGNRULEWIDTH argument, you cannot specify the SPACING argument.

Type: Float

ITERATE

Creates an array of the PATH, RECT, POLYGON, or VIA geometry, as specified by the given step pattern. ITERATE specifications simplify the definitions of cover macros. The syntax for *stepPattern* is defined as follows:

DO *numX* BY *numY* STEP *spaceX spaceY*

numX Specifies the number of columns of points.

numY Specifies the number of rows of points.

spaceX spaceY Specifies the spacing, in distance units, between the columns and rows of points.

LAYER *layerName*

Specifies the layer on which to place the geometry.

PATH *pt*

Creates a path between the specified points, such as *pt1 pt2 pt3*. The path automatically extends the length by half of the current *width* on both end points to form a rectangle. You can

LEF/DEF 5.4 Language Reference

LEF Syntax

also specify a path with a single coordinate, in which case a square whose side is equal to the current *width* is placed with its center at *pt*.

<code>POLYGON <i>pt pt pt pt</i></code>	Specifies a sequence of at least four points to generate a polygon geometry. The polygon must be orthogonal (all edges parallel to the x or y axis), so each successive point must vary only one coordinate. Each <code>POLYGON</code> statement defines a polygon generated by connecting each successive point, and by connecting the first and last points.
<code>RECT <i>pt pt</i></code>	Specifies a rectangle, where the two points specified are opposite corners of the rectangle. There is no functional difference between a geometry specified using <code>PATH</code> and a geometry specified using <code>RECT</code> .
<code>SPACING <i>minSpacing</i></code>	Specifies the minimum spacing allowed between this particular obstruction or pin and any other shape. This overrides the normal <code>LAYER</code> -based spacing rules. If you specify the <code>SPACING</code> argument, you cannot specify the <code>DESIGNRULEWIDTH</code> argument.
<code>VIA <i>pt viaName</i></code>	Specifies the via to place and the placement location.
<code>WIDTH <i>width</i></code>	Specifies the width that the <code>PATH</code> statements use. If you do not specify the argument, the default width for that layer is used. When you specify a width, that width remains in effect until the next <code>WIDTH</code> or <code>LAYER</code> statement. When another <code>LAYER</code> statement is given, the <code>WIDTH</code> is automatically reset to the default width for that layer.

Examples

The following example shows how to define a set of geometries, first by using `ITERATE` statements, then by using individual `PATH`, `VIA` and `RECT` statements.

The following two sets of statements are equivalent:

```
PATH ITERATE 532.0 534 1999.2 534
DO 1 BY 2 STEP 0 1446 ;
VIA ITERATE 470.4 475 VIABIGPOWER12
DO 2 BY 2 STEP 1590.4 1565 ;
RECT ITERATE 24.1 1.5 43.5 16.5
DO 2 BY 1 STEP 20.0 0 ;
```

LEF/DEF 5.4 Language Reference

LEF Syntax

```
PATH 532.0 534 1999.2 534 ;
PATH 532.0 1980 1999.2 1980 ;
VIA 470.4 475 VIABIGPOWER12 ;
VIA 2060.8 475 VIABIGPOWER12;
VIA 470.4 2040 VIABIGPOWER12;
VIA 2060.8 2040 VIABIGPOWER12;
RECT 24.1 1.5 43.5 16.5 ;
RECT 44.1 1.5 63.5 16.5 ;
```

Macro Obstruction Statement

```
[OBS
  { LAYER layerName [SPACING minSpacing | DESIGNRULEWIDTH value] ;
    [WIDTH width ;]
    { PATH pt ... ;
      | PATH ITERATE pt ... stepPattern ;
      | RECT pt pt ;
      | RECT ITERATE pt pt stepPattern ;
      | POLYGON pt pt pt pt ... ;
      | POLYGON ITERATE pt pt pt pt ... stepPattern ;
    } ...
    | VIA pt viaName ;
    | VIA ITERATE pt viaName stepPattern ;
  } ...
END]
```

Defines a set of obstructions (also called blockages) on the macro. You specify obstruction geometries using the layer geometry syntax. See [“Layer Geometries”](#) on page 44 for syntax information.

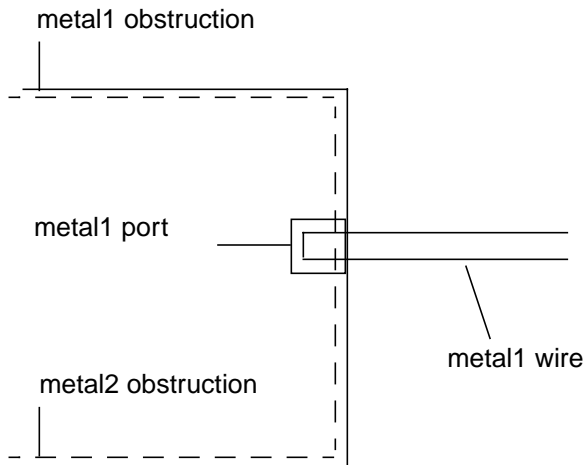
Normally, obstructions block routing, except for when a pin port overlaps an obstruction (a port geometry overrules an obstruction). For example, you can define a large rectangle for a metal1 obstruction and have metal1 ports in the middle of the obstruction. The port can still be accessed by a via, if the via is entirely inside the port.

In [Figure 1-1](#) on page 47, the router can only access the metal1 port from the right. If the metal2 obstruction did not exist, the router could connect to the port with a metal12 via, as long as the metal1 part of the via fit entirely inside the metal1 port.

LEF/DEF 5.4 Language Reference

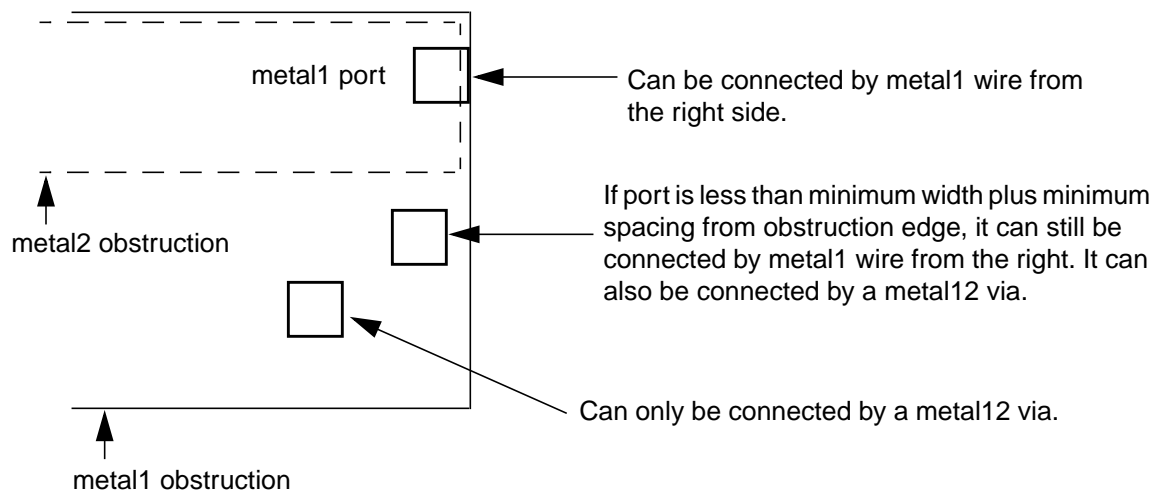
LEF Syntax

Figure 1-1



Routing can also connect to such a port on the same layer if the routing does not cross any obstruction by more than a distance of the total of minimum width plus minimum spacing before reaching the pin. This is because the port geometry is known to be "real," and any obstruction less than a distance of minimum width plus minimum spacing away from the port is not a real obstruction. If the pin is more than minimum width plus minimum spacing away from the obstruction edge, the router can only route to the pin from the layer above or below using a via (see [Figure 1-2](#) on page 47).

Figure 1-2



If a port is on the edge of the obstruction, a wire can be routed to the port without violations. Pins that are partially covered with obstructions or in apparent violation with nearby

LEF/DEF 5.4 Language Reference

LEF Syntax

obstructions can limit routing options. Even though the violations are not real, the router assumes they are. In these cases, extend each obstruction to cover the pin. The router then accesses the pin as described above.

Benefits of Combining Obstructions

Significant routing time can be saved if obstructions are simplified. Especially in metal1, construct obstructions so that free tracks on the layer are accessible to the router. If most of the routing resource is obstructed, simplify the obstruction modeling by combining small obstructions into a single large obstruction. For example, use the bounding box of all metal1 objects in the cell, rather than many small obstructions, as the bounding box of the obstruction.

You must be sure to model via obstructions over the rest of the cell properly. A single, large cut12 obstruction over the rest of the cell can do this in some cases, as when metal1 resource exists within the cell outside the power buses.

Rectilinear Blocks

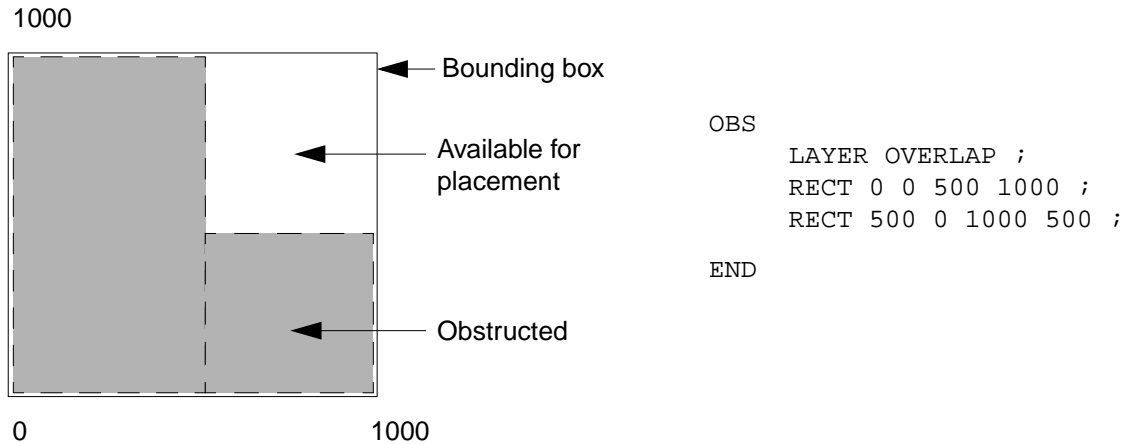
Normally, footprint descriptions in LEF are rectangular. However, it is possible to describe rectilinear footprints using an overlap layer. The overlap layer is defined specifically for this purpose and does not contain any routing.

Describe a rectilinear footprint by setting the `SIZE` of the macro as a whole to a rectangular bounding box, then defining obstructions within the bounding box on the overlap layer. The obstructions on the overlap layer indicate areas within the bounding box which no other macro should overlap. The obstructions should completely cover the rectilinear shape of the macro, but not the portion of the bounding box that might overlap with other macros during placement.

LEF/DEF 5.4 Language Reference

LEF Syntax

Note: Specify the overlaps for the macro using the `OBS` statement. To do this, specify a layer of type `OVERLAP` and then give the overlap geometries, as shown in the following figure.



Macro Pin Statement

```
[PIN pinName
  [TAPERRULE ruleName ;]
  [FOREIGN foreignPinName [STRUCTURE [pt [orient] ] ] ;]
  [LEQ pinName ;] ...
  [DIRECTION {INPUT | OUTPUT [TRISTATE] | INOUT | FEEDTHRU} ;]
  [USE { SIGNAL | ANALOG | POWER | GROUND | CLOCK } ;]
  [SHAPE {ABUTMENT | RING | FEEDTHRU} ;]
  [MUSTJOIN pinName ;]
  {PORT
    [CLASS {NONE | CORE} ;]
    {layerGeometries} ...
  }
  END} ...
  [PROPERTY propName propVal ;] ...
  [ANTENNAPARTIALMETALAREA value [LAYER layerName] ;] ...
  [ANTENNAPARTIALMETALSIDEAREA value [LAYER layerName] ;] ...
  [ANTENNAPARTIALCUTAREA value [LAYER layerName] ;] ...
  [ANTENNAGATEAREA value [LAYER layerName] ;] ...
  [ANTENNADIFFAREA value [LAYER layerName] ;] ...
  [ANTENNAMAXAREACAR value LAYER layerName ;]
  [ANTENNAMAXSIDEAREACAR value LAYER layerName ;]
  [ANTENNAMAXCUTCAR value LAYER layerName ;]
END pinName]
```

Defines pins for the macro. `PIN` statements must be included in the LEF specification for each macro. All pins, including VDD and VSS, must be specified. The first pin listed becomes the first pin in the database. List the pins in the following order:

- Netlist pins, including inout pins, output pins, and input pins

LEF/DEF 5.4 Language Reference

LEF Syntax

- Power and ground pins
- Mustjoin pins

`ANTENNADIFFAREA value [LAYER layerName]`

Specifies the diffusion (diode) area, in micron-squared units, to which the pin is connected on a layer. If you do not specify a layer name, the value applies to all layers. For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

`ANTENNAGATEAREA value [LAYER layerName]`

Specifies the gate area, in micron-squared units, to which the pin is connected on a layer. If you do not specify a layer name, the value applies to all layers. For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

`ANTENNAMAXAREACAR value LAYER layerName`

For hierarchical process antenna effect calculation, specifies the maximum cumulative area ratio value on the specified *layerName*, using the metal area at or below the current pin layer, excluding the pin area itself. This is used to calculate the actual cumulative antenna ratio on the pin layer, or the layer above it. For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

`ANTENNAMAXCUTCAR value LAYER layerName`

For hierarchical process antenna effect calculation, specifies the maximum cumulative antenna ratio value on the specified *layerName*, using the cut area at or below the current pin layer, excluding the pin area itself. This is used to calculate the actual cumulative antenna ratio for the cuts above the pin layer. For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

`ANTENNAMAXSIDEAREACAR value LAYER layerName`

For hierarchical process antenna effect calculation, specifies the maximum cumulative antenna ratio value on the specified *layerName*, using the metal side wall area at or below the current pin layer, excluding the pin area itself. This is used to calculate the actual cumulative antenna ratio on the pin layer or the layer above it. For information on process antenna

LEF/DEF 5.4 Language Reference

LEF Syntax

calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

ANTENNAPARTIALCUTAREA *value* [LAYER *layerName*]

Specifies the partial cut area above the current pin layer and inside the macro cell on the layer. For a hierarchical design, only the cut layer above the I/O pin layer is needed for partial antenna ratio calculation. If you do not specify a layer name, the value applies to all layers. For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

ANTENNAPARTIALMETALAREA *value* [LAYER *layerName*]

Specifies the partial metal area connected directly to the I/O pin and the inside of the macro cell on the layer. For a hierarchical design, only the same metal layer as the I/O pin, or the layer above it, is needed for partial antenna ratio calculation. If you do not specify a layer name, the value applies to all layers. For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

Note: Metal area is calculated by adding the pin's geometric metal area and the ANTENNAPARTIALMETALAREA value.

ANTENNAPARTIALMETALSIDEAREA *value* [LAYER *layerName*]

Specifies the partial metal side wall area connected directly to the I/O pin and the inside of the macro cell on the layer. For a hierarchical design, only the same metal layer as the I/O pin or the layer above is needed for partial antenna ratio calculation. If you do not specify a layer name, the value applies to all layers. For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

CLASS {NONE | CORE}

Specifies whether or not the port is a core port. A core port is only used on power and ground I/O pads. The core port indicates which power or ground port to connect to a core ring for the chip (inside the I/O pads).

DIRECTION {INPUT | OUTPUT [TRISTATE] | INOUT | FEEDTHRU}

Specifies the pin type.

Default: INPUT

Value: Specify one of the following:

INPUT Pin that accepts signals coming into the cell.

LEF/DEF 5.4 Language Reference

LEF Syntax

OUTPUT [TRISTATE]	Pin that drives signals out of the cell. The optional TRISTATE argument indicates tristate output pins for ECL designs.
INOUT	Pin that can accept signals going either in or out of the cell.
FEEDTHRU	

FOREIGN *foreignPinName*

Specifies which foreign system pin name within the macro GDSII structure or which GDSII structure itself to use to extract pin port geometries from, and output to, when an instance of this macro is placed and this pin is used.

layerGeometries

Defines port geometries for the pin. You specify port geometries using layer geometries syntax. See [“Layer Geometries”](#) on page 44 for syntax information.

LEQ *pinName*

Specifies that the pin being defined should be logically equivalent to the previously defined *pinName*. All the LEQ pins equivalent to the first pin must be defined immediately after the first pin. LEQ pins must be defined contiguously.

MUSTJOIN *pinName*

Specifies the name of another pin in the cell that must be connected with the pin being defined. Mustjoin pins provide connectivity that must be made on the routing layers. In the LEF file, each pin referred to must be defined before the referring pin. The remaining mustjoin pins in the set do not need to be defined contiguously.

Mustjoin pins have the following restrictions:

- An mustjoin pin cannot be a LEQ pin.
- A set of mustjoin pins cannot have more than one schematic pin.
- Nonschematic mustjoin pins must be defined after all other pins.
- LEQ macros can have a different number and order of nonschematic mustjoin pins.

LEF/DEF 5.4 Language Reference

LEF Syntax

Schematic and nonschematic mustjoin pins are handled in slightly different ways. For schematic mustjoin pins, the pins are added to the pin set for the (unique) net associated with the ring for each component instance of the macro. The net is routed in the usual manner, and routing data for the mustjoin pins are included in routing data for the net.

The mustjoin routing is not necessarily performed before the rest of the net. Timing relations should not be given for mustjoin pins, and internal mustjoin routing is modeled as lumped capacitance at the schematic pin.

Nonschematic mustjoin pin sets get routed in the usual manner. However, when the DEF file is outputted, routing data is reported in the NETS section of the file as follows:

```
MUSTJOIN compName pinName + regularWiring ;
```

Here, *compName* is the component and *pinName* is an arbitrary pin in the set. You can also use the preceding to input prewiring for the mustjoin pin, using FIXED or COVER.

PIN *pinName*

Specifies the name for the library pin.

PORT

Starts a pin port statement, which defines a collection of geometries that are electrically equivalent points. A pin can have multiple ports. Routers connect to any of the member geometries of a port, without affecting the electrical performance of the circuit.

PROPERTY *propName propVal*

Specifies a numerical or string value for a pin property defined in the PROPERTYDEFINITIONS statement. The *propName* you specify must match the *propName* listed in the PROPERTYDEFINITIONS statement.

SHAPE

Specifies a pin with special connection requirements because of its shape.

Value: Specify one of the following:

ABUTMENT

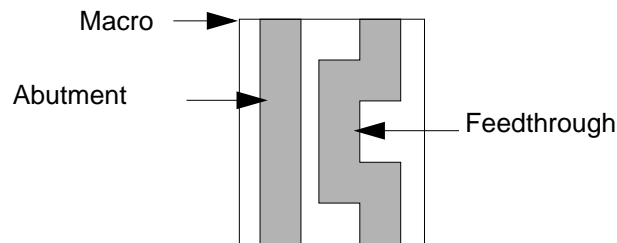
Pin that goes straight through cells with a regular shape and connects to pins on adjoining cells without routing.

LEF/DEF 5.4 Language Reference

LEF Syntax

RING	Pin on a large block that forms a ring around the block to allow connection to any point on the ring. Cover macro special pins also typically have shape RING.
FEEDTHRU	Pin with an irregular shape with a jog or neck within the cell.

The following figure shows an example of an abutment and a feedthrough pin.



Note: For Silicon Ensemble, when you define feedthrough and abutment pins for use with `SROUTE FOLLOWPINS`, you must do the following:

- Feedthrough pin widths must be the same on both edges and consistent with the routing width used with the `SROUTE FOLLOWPINS` command.
- Feedthrough pin centers on both edges must align for successful routing.
- Power pins in fork shapes must be represented in two ports and be defined as a feedthrough shape. In most other cases, power pin geometries do not represent more than one port.
- An abutment pin must have at least one geometric rectangle with layer and width consistent with the values specified in the `SROUTE FOLLOWPINS` command.

`STRUCTURE [pt [orient]]`

Includes all or part of the pin physical data in a separate GDSII structure, rather than the macro GDSII structures. This is normally used for LEQ pins; and it allows the different physical implementations for each LEQ pin to be selectable when GDSII is output, based upon which pin is used. The pin port geometries are the union of those found in the macro GDSII structure (by pin name texting) and the geometries in the pin GDSII structure (also texted).

pt specifies the pin origin (lower left corner when the macro is in north orientation) offset from the foreign origin.

LEF/DEF 5.4 Language Reference

LEF Syntax

orient specifies the orientation of the foreign cell when the macro is in north orientation.

For Silicon Ensemble, the calculation of absolute location for the foreign reference in `OUTPUT GDSII` is affected by component placement location, component orientation, and the pin foreign coordinate offset, but not by the pin foreign orientation.

`TAPERRULE ruleName` Specifies the nondefault rule to use when tapering wires to the pin.

`USE {ANALOG | CLOCK | GROUND | POWER | SIGNAL}`
Specifies how the pin is used. Pin use is required for timing analysis.

Value: Specify one of the following:

ANALOG	For Integration Ensemble only, pin is used for analog connectivity.
CLOCK	Pin is used for clock net connectivity.
GROUND	Pin is used for connectivity to the chip-level ground distribution network.
POWER	Pin is used for connectivity to the chip-level power distribution network.
SIGNAL	Pin is used for regular net connectivity.

Macro Pin Considerations

Note the following when defining macro pins:

- LEQ pins must be identical for EEQ or LEQ macros.
- A given macro can have more than one set of LEQ pins.
- A set of LEQ pins cannot have more than one schematic pin (that is, a component pin connected to a net in DEF).
- You must manually select which pins to connect in the netlist.

Manufacturing Grid

`[MANUFACTURINGGRID value ;]`

LEF/DEF 5.4 Language Reference

LEF Syntax

Defines the manufacturing grid for the design. The manufacturing grid is used for geometry alignment. When specified, shapes and cells are placed in locations that snap to the manufacturing grid.

value Specifies the value for the manufacturing grid, in microns.
value must be a positive number.
Type: Float

Names Case Sensitive

```
NAMESCASESENSITIVE {ON | OFF} ;
```

Specifies if LEF data is case sensitive.

For Silicon Ensemble, the case sensitivity of the first `INPUT LEF` command is determined by the value of `NAMESCASESENSITIVE`. The `NAMESCASESENSITIVE` statement must be on the first noncommented line after the `VERSION` statement. If it is not in the correct position, `INPUT LEF` checks the setting of the environmental variable `Input.Lef.Names.Case.Sensitive`.

Nondefault Rule

```
[NONDEFAULTRULE rulename1
  {LAYER layerName
    WIDTH width ;
    SPACING minSpacing ;
    [WIREEXTENSION value ;]
    [RESISTANCE RPERSQ value ;]
    [CAPACITANCE CPERSQDIST value ;]
    [EDGECAPACITANCE value ;]
  END layerName} ...
  {VIA viaStatement} ...
  [SPACING
    [SAMENET layerName layerName minSpace [STACK] ;] ...
  END SPACING]
  [PROPERTY propName propValue ;] ...
END]
```

Defines the wiring width, design rule spacing, and via size to regular (signal) nets. These characteristics must be equal to or greater than those characteristics for the default regular routing. Each succeeding nondefault rule on the same layer must be equal to or greater than the previous nondefault rule. For example:

LEF/DEF 5.4 Language Reference

LEF Syntax

```
Nondefault rule 1 (NDrule1) for M1 >= Default rule for all layers
NDrule2 for M1 >= NDrule1 for M1
NDrule3 for M1 >= NDrule2 for M1
NDrule4 for M1 >= NDrule3 for M1
```

You do not need to define cut layers for the nondefault rule. You can have up to 30 nondefault rules for a design.

Note: Use the `VIA` statement to define vias for nondefault wiring.

`CAPACITANCE CPERSQDIST` *value*

Specifies the capacitance for each square unit, in picofarads per square micron. This is used to model wire-to-ground capacitance.

`EDGECAPACITANCE` *value*

Specifies a floating-point value of peripheral capacitance, in picofarads per micron. The place-and-route tool uses this value in two situations:

- Estimating capacitance before routing
- Calculating segment capacitance after routing

For the second calculation, the tool uses *value* only if you set layer thickness, or layer height, to 0. In this situation, the peripheral capacitance is used in the following formula:

segment capacitance = (layer capacitance per square x segment width x segment length) + (peripheral capacitance x 2 (segment width + segment length))

For Silicon Ensemble, if you do not specify a value for `EDGECAPACITANCE`, the values of the `Timing.Fringcap.layernum` environmental variables are used instead.

`LAYER` *layerName*

Specifies the name of the layer.

`NONDEFAULTRULE` *rulename1*

Specifies the name for the nondefault rule.

`PROPERTY` *propName propVal*

Specifies a numerical or string value for a nondefault rule property defined in the `PROPERTYDEFINITIONS` statement.

LEF/DEF 5.4 Language Reference

LEF Syntax

The *propName* you specify must match the *propName* listed in the PROPERTYDEFINITIONS statement.

RESISTANCE RPERSQ *value*

Specifies the resistance for a square of wire, in ohms per square micron.

Type: Float

SPACING

Defines same-net spacing for the nondefault rule. This argument uses the same syntax as the same-net spacing statement. For syntax information, see [“Same-Net Spacing”](#) on page 62.

SPACING *minSpacing*

Specifies the recommended minimum spacing of routes using this NONDEFAULTRULE to other geometries. If the spacing is given, it must be at least as large as the foundry minimum spacing rules defined in the LAYER definitions. Routers should attempt to meet this recommended spacing rule; however, the spacing rule can be relaxed to the foundry spacing rules along some parts of the wire if the routing is very congested, or it is difficult to reach a pin.

Adding extra space to a nondefault rule allows a designer to reduce cross-coupling capacitance and noise, but a clean route with no actual foundry spacing violations will still have precedence.

VIA *viaStatement*

Defines a via. You must define vias for each nondefault rule. You define nondefault vias using the same syntax as default vias. For syntax information, see [“Via”](#) on page 67. All vias, default and nondefault, must have unique via names.

If you define more than one via for a rule, the router chooses which via to use.

By omitting the WIDTH statement (specified below), you can also create a nondefault rule that covers all vias that do not have via widths explicitly defined.

WIDTH *width*

Specifies the nondefault routing width. The nondefault routing width for the first rule name must be wider than or equal to the default wiring. The nondefault width for each successive rule name must be wider than or equal to the width of the previous rule name.

LEF/DEF 5.4 Language Reference

LEF Syntax

WIREEXTENSION *value* Specifies the distance by which wires are extended at vias. Enter 0 to specify no extension. Values other than 0 must be at least half of the routing width for the layer or larger, as defined in the nondefault rule.
Default: Nondefault wires have zero extension
Type: Float

Examples

```
NONDEFAULTRULE RULE1
  LAYER M1
    WIDTH 10.0 ;
    SPACING 2.2 ;
  END M1

  LAYER M2
    WIDTH 10.0 ;
    SPACING 2.2 ;
  END M2

  LAYER M3
    WIDTH 11.0 ;
    SPACING 3.2 ;
  END M3

  VIA nd1VIA12
    FOREIGN IN1X ;
    RESISTANCE 0.2 ;
    LAYER M1 ;
    RECT -3 -3 3 3 ;
    LAYER V1 ;
    RECT -1.0 -1.0 1.0 1.0 ;
    LAYER M2 ;
    RECT -3 -3 3 3 ;
  END nd1VIA12

  VIA nd1VIA23
    LAYER M3 ;
    RECT -2.2 -2.2 2.2 2.2 ;
    LAYER V2 ;
    RECT -0.8 -0.8 0.8 0.8 ;
    LAYER M2 ;
    RECT -2.0 -2.0 2.0 2.0 ;
  END nd1VIA23

  SPACING
    SAMENET
    CUT01 M1 0.1 STACK ;
  END SPACING

END RULE1
```

LEF/DEF 5.4 Language Reference

LEF Syntax

In the following example, `RULE1` uses a wire extension of 70 on layer *M1* instead of the default extension of 0.

```
NONDEFAULTRULE RULE1
  LAYER M1
  WIDTH 120 ;
  WIREEXTENSION 70 ;
  .....
  END M1
  .....
END
```

Because no wire extension is specified in `RULE2`, the rule defaults to 0.

```
NONDEFAULTRULE RULE2
  LAYER M1
  WIDTH 130 ;
  .....
  END M1
  .....
END
```

No Wire Extension

```
[NOWIREEXTENSIONATPIN {ON | OFF} ;]
```

Specifies whether to extend wires at pins. By default (`OFF`), wires are extended one-half of their width at pins.

Property Definitions

```
[PROPERTYDEFINITIONS
  [objectType propName propType [RANGE min max]
   [value | "stringValue"]
  ;] ...
END PROPERTYDEFINITIONS]
```

Lists all properties used in the LEF file. You must define properties in the `PROPERTYDEFINITIONS` statement before you can refer to them in other sections of the LEF file.

<i>objectType</i>	Specifies the object type being defined. You can define properties for the following object types:
	LAYER

LEF/DEF 5.4 Language Reference

LEF Syntax

LIBRARY
MACRO
NONDEFAULTRULE
PIN
VIA
VIARULE

<i>propName</i>	Specifies a unique property name for the object type.
<i>propType</i>	Specifies the property type for the object type. You can specify one of the following property types: INTEGER REAL STRING
RANGE <i>min max</i>	Limits real number and integer property values to a specified range. That is, the value must be greater than or equal to <i>min</i> and less than or equal to <i>max</i> .
<i>value</i> " <i>stringValue</i> "	Assigns a numeric value or a name to a LIBRARY object type. Note: Assign values to other properties in the section of the LEF file that describes the object to which the property applies.

Examples

The following example shows library via and macro property definitions.

```
PROPERTYDEFINITIONS
  LIBRARY versionNum INTEGER 12;
  LIBRARY title STRING "Cadence96";

  VIA count INTEGER RANGE 1 100;

  MACRO weight REAL RANGE 1.0 100.0;
  MACRO type STRING;

END PROPERTYDEFINITIONS
```

LEF/DEF 5.4 Language Reference

LEF Syntax

Same-Net Spacing

```
[SPACING  
    [SAMENET layerName layerName minSpace [STACK] ;] ...  
END SPACING]
```

Defines the same-net spacing rules. Same-net spacing rules determine minimum spacing between geometries in the same net and are only required if same-net spacing is smaller than different-net spacing, or if vias on different layers have special stacking rules. These specifications are used for design rule checking by the routing and verification tools.

Spacing is the edge-to-edge separation, both orthogonal and diagonal. Same-net spacing rules are less restrictive than different-net spacing rules.

Note: If SAMENET rules are not defined, via stacking is legal by default.

layerName layerName Specifies the names of the layers for which the same-net spacing rule applies. You can specify spacing rules for routing layers and cut layers. For a routing layer, the same-net spacing rule is defined by specifying the same layer name twice.

minSpace Specifies the minimum spacing.

STACK Allows stacked vias at a routing layer.

Same-Net Spacing Rule Examples

The following example sets the minimum same-net spacing for layer *M1*:

```
SAMENET M1 M1 1.5 ;
```

This statement is only needed if the same-net spacing is different than the different-net spacing already in the *M1* LAYER section.

The following example defines spacing for cut layers. The statement sets the same-net spacing to 2.3 microns (also in the xy plane) for pairs of features, where one geometry is on CUT01 and the other geometry is on CUT12, and disables stacked (overlapped) vias.

```
SAMENET CUT01 CUT12 2.3 ;
```

You can allow stacked vias at a routing layer by including the STACK keyword in the same-net specification for that layer. For example, the following specification allows a 1-2 via and a 2-3 via to be stacked at the same point in the design. If they do not exactly align, the same-net spacing rule of 1.5 microns is applied.

```
SAMENET CUT12 CUT23 1.5 STACK ;
```

LEF/DEF 5.4 Language Reference

LEF Syntax

If you want to allow arbitrary stacking between vias, you can either use the default behavior by not specifying a SAMENET rule, or you can do it explicitly using the following statement:

```
SAMENET via12 via23 0.0 STACK ;
```

Nondefault Rules and Same-Net Spacing

NONDEFAULT rules inherit the default SPACING SAMENET rules. However, if the NONDEFAULT rule has its own SAMENET rules, these rules override the default values.

For example, no SAMENET rules at all allows stacking and overlap for default and nondefault routes. However, the following default:

```
SAMENET cut12 cut23 0.1 ;
```

turns off via12 and via23 stacking for all default and nondefault routes, and requires the vias to be separated by 0.1 microns.

A default SPACING section with

```
SAMENET cut12 cut23 0.1 ;
```

followed by a NONDEFAULT rule with

```
SAMENET cut12 cut23 0.1 STACK ;
```

allows an exactly aligned stacked via for the nondefault routes, but does not allow stacking for the nondefault routes.

Site

```
SITE siteName  
  CLASS {PAD | CORE} ;  
  [SYMMETRY {X | Y | R90} ... ;]  
  SIZE width BY height ;  
  
END siteName
```

Defines a placement site in the design. A placement site gives the placement grid for a family of macros, such as I/O, core, block, analog, digital, short, tall, and so forth.

CLASS {PAD | CORE} Specifies whether the site is an I/O pad site or a core site.

SITE *siteName* Specifies the name for the placement site.

SIZE *width* BY *height* Specify the dimensions of the site in normal (or north) orientation, in microns.

LEF/DEF 5.4 Language Reference

LEF Syntax

`SYMMETRY {X | Y | R90}`

Specifies whether the site in normal orientation is symmetric about the x axis, about the y axis, or when rotated 90 degrees. If the `SYMMETRY` statement is not given, the system assumes that the site is completely asymmetric.

Placement sites are usually symmetric in `X` and `Y` to allow greater flexibility in specifying row placement orientation. However, they are not symmetric in `R90`.

Units

```
[UNITS
  [TIME NANOSECONDS convertFactor ;]
  [CAPACITANCE PICO FARADS convertFactor ;]
  [RESISTANCE OHMS convertFactor ;]
  [POWER MILLIWATTS convertFactor ;]
  [CURRENT MILLIAMPS convertFactor ;]
  [VOLTAGE VOLTS convertFactor ;]
  [DATABASE MICRONS LEFconvertFactor ;]
  [FREQUENCY MEGAHERTZ convertFactor ;]
END UNITS]
```

Defines the units of measure in LEF. The values tell you how to interpret the numbers found in the LEF file. Units are fixed with a *convertFactor* for all unit types, except database units and capacitance. For more information, see [“Convert Factors”](#) on page 65. Currently, other values for *convertFactor* appearing in the `UNITS` statement are ignored; these values are intended for future use.

The `UNITS` statement is optional and, when used, must precede the `LAYER` statements.

`CAPACITANCE PICO FARADS convertFactor`

Interprets one LEF capacitance unit as 1 picofarad.

`CURRENT MILLIAMPS convertFactor`

Interprets one LEF current unit as 1 milliamp.

`DATABASE MICRONS LEFconvertFactor`

Interprets one LEF distance unit as multiplied when converted into database units.

If you omit the `DATABASE MICRONS` statement, a default value of 100 is recorded as the *LEFconvertFactor* in the database. In this case, one micron would equal 100 database units.

LEF/DEF 5.4 Language Reference

LEF Syntax

FREQUENCY MEGAHERTZ *convertFactor*

Interprets one LEF frequency unit as 1 megahertz.

POWER MILLIWATTS *convertFactor*

Interprets one LEF power unit as 1 milliwatt.

RESISTANCE OHMS *convertFactor*

Interprets one LEF resistance unit as 1 ohm.

TIME NANOSECONDS *convertFactor*

Interprets one LEF time unit as 1 nanosecond.

VOLTAGE VOLTS *convertFactor*

Interprets one LEF voltage unit as 1 volt.

Database Units Information

Database precision is relative to Standard International (SI) units. LEF values are converted to integer values in the library database as follows.

SI unit	Database precision
1 nanosecond	= 1,000 DBUs
1 picofarad	= 1,000,000 DBUs
1 ohm	= 10,000 DBUs
1 milliwatt	= 10,000 DBUs
1 milliamp	= 10,000 DBUs
1 volt	= 1,000 DBUs

Convert Factors

Currently, LEF only supports values of 100, 200, 1000, and 2000 for *LEFconvertFactor*. The following table illustrates the conversion of LEF distance units into database units.

LEFconvertFactor	LEF	Database Units
100	1 micron	100
200	1 micron	200
1000	1 micron	1000

LEF/DEF 5.4 Language Reference

LEF Syntax

LEFconvertFactor	LEF	Database Units
2000	1 micron	2000

The DEF database precision cannot be more precise than the LEF database precision. This means the DEF convert factor must always be less than or equal to the LEF convert factor. The following table shows the valid pairings of the LEF convert factor and the corresponding DEF convert factor.

LEFconvertFactor	DEFconvertFactor
100	100
200	100
200	200
1000	100
1000	1000
2000	100
2000	2000

Note: For Silicon Ensemble, the LEF convert factor must be the same as the DEF convert factor.

Any other values or combinations of the LEF convert factor and DEF convert factor are not supported in the `DATABASE MICRONS` statement.

An incremental LEF should have the same value as a previous LEF. An error message warns you if an incremental LEF has a different value than what is recorded in the database.

Use Min Spacing

```
[USEMINSPACING OBS { ON | OFF } ;]
```

```
[USEMINSPACING PIN { ON | OFF } ;]
```

Defines how minimum spacing is calculated for obstruction (blockage) and pin geometries.

OBS Specifies the minimum spacing applies to obstruction geometries.

ON | OFF Specifies how to calculate minimum spacing. If you specify ON (the default for obstruction), the spacing is the minimum layer

LEF/DEF 5.4 Language Reference

LEF Syntax

spacing. If you specify `OFF` (the default for pins), the spacing is the range spacing that corresponds to the minimum width (in x and y directions) of the pin or obstruction.

PIN Specifies the minimum spacing applies to pin geometries.

Version

```
VERSION number ;
```

Specifies which version of the LEF syntax is being used. *number* is a string of the form *major.minor.subMinor*, such as 5.3.1.

Via

```
VIA viaName
    [DEFAULT]
    [TOPOFSTACKONLY]
    [FOREIGN foreignCellName [pt [orient]]]
    [RESISTANCE value ;]
    {LAYER layerName ;
      {RECT pt pt ;} ...} ...
    [PROPERTY propName propName ;] ...

END viaName
```

Defines vias for nondefault regular wiring.

Note: Use the `VIARULE` section to define special wiring. Use the `VIARULE GENERATE` section to cover special wiring that is not explicitly defined in the `VIARULE` section.

DEFAULT Identifies the via as the default via between the defined layers. If you define more than one default via for a layer pair, the router will chose which via to use. You can define up to 10 vias for each layer pair that might be directly connected. You must define default vias between metal1 and masterslice layers if there are pins on the masterslice layers.

Note: Default vias have exactly three layers used: a cut layer, and two layers that touch the cut layer (routing or masterslice). The cut layer must be between the two routing or masterslice layer rectangles.

```
FOREIGN foreignCellName [pt [orient]]
```

Specifies the foreign (GDSII) structure name to use when

LEF/DEF 5.4 Language Reference

LEF Syntax

placing an instance of the via. Only one foreign declaration is allowed for a via. The optional *pt* coordinate specifies the cell origin (normally, the via center) offset relative to the foreign origin. *orient* specifies the orientation of the foreign cell when the macro is in north orientation.

For vias used in any routing, via coordinates must be consistent with their location at the intersection of the wires. Vias referenced in pins or obstructions must have the GDSII origin at the center of the via.

LAYER *layerName*

Specifies the layer on which to create the rectangles that make up the via. Normal vias have exactly three layers used: a cut layer, and two layers that touch the cut layer (routing or masterslice).

In the past, it was possible to create a via with just a cut layer, but this is not recommended because many tools cannot support this style of via. This type of via was only used to create obstructions, and does not appear in real routing data. Cut layer obstructions should be put directly into the MACRO OBS section instead of using a "cut layer" via.

WARNING: Vias can be rectilinear rather than square, but some routers cannot orient vias; they all stay in normal orientation in the layout. Define vias of the proper shape for each direction needed. Be careful of non-square vias in components that can be rotated, because the vias cannot be oriented. Define components with appropriate vias.

PROPERTY *propName propVal*

Specifies a numerical or string value for a via property defined in the PROPERTYDEFINITIONS statement. The *propName* you specify must match the *propName* listed in the PROPERTYDEFINITIONS statement.

RECT *pt pt*

Specify the geometries that make up the via. For vias used only in macros or pins, reference locations and rectangle coordinates must be consistent.

RESISTANCE *value*

Specifies the lumped resistance of the contact cuts, given as the resistance per via.

LEF/DEF 5.4 Language Reference

LEF Syntax

TOPOFSTACKONLY Prevents the router from using the via except for via stacking. When defining stacked vias, use **TOPOFSTACKONLY** to instruct the router to use this via (rather than the default via) to satisfy minimum area rules.

For a *metal1* to *metal5* stack, the **TOPOFSTACKONLY** via will be used for all except the bottom via. Thus, the bottom routing layer of a **TOPOFSTACKONLY** via should satisfy the minimum area rules.

viaName Specifies the name for the via.

Preventing Vias in Specific Locations

To prevent the use of vias at particular locations, specify a via with a cut layer, but no top or bottom routing layers. You can place vias as required in a design using DEF to specify their physical locations in the netlist or placed within library descriptions of macros. The cut layer prevents placement of authentic vias at these locations.

Because the routing layers do not exist, these vias are not output as GDSII geometries. You can forbid via locations by specifying a blockage in a macro description on the cut layer only.

Examples

The following example defines a **FOREIGN** argument for via B.

```
VIA B FOREIGN B -2 -3 ;
```

The negative offset specifies that the GDSII structure origin should be above and to the right of the via origin. For example, for a 2-by-2 via, with (0 0) to (2 2) coordinates in GDSII, the **FOREIGN** statement uses (1 1) to put the origin at the center.

The following example defines a three-layer design with the **TOPOFSTACKONLY** keyword. To satisfy the minimum-area rule requirement, you specify three vias: the default via and two overhang vias to accommodate two different directions. This provides the router the flexibility to choose the most suitable via during routing.

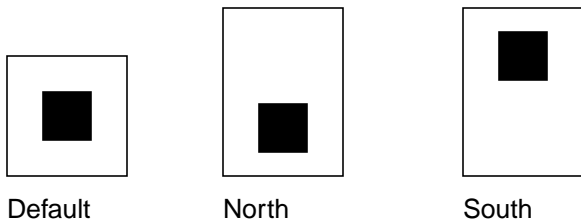
The following example defines default **Via23** with layer *Metal2*, *Metal3*, and *Cut23* contact cut.

```
VIA Via23 DEFAULT
LAYER Metal2 ;
RECT -0.7 -0.7 0.7 0.7 ;
LAYER Cut23 ;
```

LEF/DEF 5.4 Language Reference

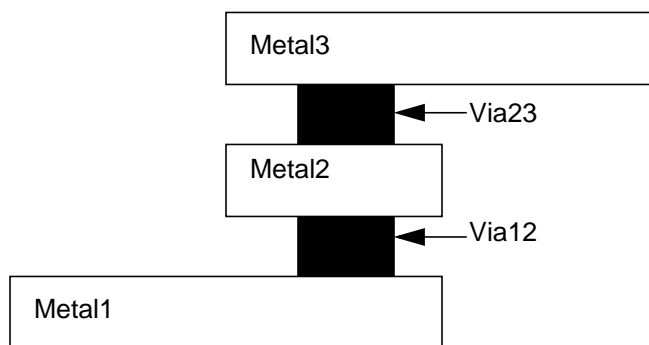
LEF Syntax

```
RECT -0.4 -0.4 0.4 0.4 ;
LAYER Metal3 ;
RECT -0.6 -0.6 0.6 0.6 ;
END Via23
```



The following examples defines `Via23_north` with `Metal2`, `Metal3`, and contact cut placed at the top. Notice that `Metal2` is longer and extends to the north.

```
VIA Via23_north DEFAULT
TOPOFSTACKONLY
LAYER Metal2 ;
RECT -0.7 -0.7 0.7 1.7 ;
LAYER Cut23 ;
RECT -0.4 -0.4 0.4 0.4 ;
LAYER Metal3 ;
RECT -0.6 -0.6 0.6 0.6 ;
END Via23_north
```



The following example defines `Via23_south` with `Metal2`, `Metal3`, and the contact cut placed at the top. Notice that `Metal2` is longer and extends to the south.

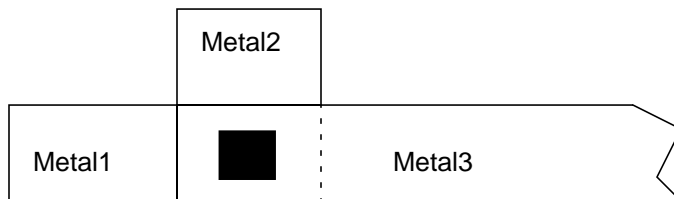
```
VIA Via23_south DEFAULT
TOPOFSTACKONLY
LAYER Metal2 ;
```

LEF/DEF 5.4 Language Reference

LEF Syntax

```
RECT -0.7 -1.7 0.7 0.7 ;
LAYER Cut23 ;
RECT -0.4 -0.4 0.4 0.4 ;
LAYER Metal3 ;
RECT -0.6 -0.6 0.6 0.6 ;
END Via23_south
```

The following figure shows a foreign offset.



Via Rule

```
VIARULE viaRuleName
  LAYER layerName ;
    DIRECTION {HORIZONTAL | VERTICAL} ;
    [WIDTH minWidth TO maxWidth ;]
  LAYER layerName ;
    DIRECTION {HORIZONTAL | VERTICAL} ;
    [WIDTH minWidth TO maxWidth ;]
  {VIA viaName ;} ...
  [PROPERTY propName propVal ;] ...
END viaRuleName
```

Defines which vias to use at the intersection of special wires of the same net. Normally, a VIARULE GENERATE statement is used to create a via for the intersection of two special wires. However, if a special case occurs that cannot be defined by the VIARULE GENERATE statement—for example, you require a via with unusual offsets or geometry—you can use the VIARULE statement to specifically describe the type of via to use at the intersection of two special wires on specific routing layers with a specific width.

You must specify appropriate via rules for all possible wire widths that can occur in special wiring.

DIRECTION {HORIZONTAL | VERTICAL}

Specifies the wire direction. If you specify a WIDTH range, the rule applies to wires of the specified DIRECTION that fall within the range. Otherwise, the rule applies to all wires of the specified DIRECTION on the layer.

LEF/DEF 5.4 Language Reference

LEF Syntax

`LAYER layerName` Specifies the routing layers for the top or bottom of the via.

`PROPERTY propName propVal`
Specifies a numerical or string value for a via rules property defined in the `PROPERTYDEFINITIONS` statement. The *propName* you specify must match the *propName* listed in the `PROPERTYDEFINITIONS` statement.

`VIA viaName`
Specifies a previously defined via to test for the current via rule. The first via in the list that can be placed at the location without design rule violations is selected. The vias must all have exactly three layers in them. The three layers must include the same routing layers as listed in the `LAYER` statements of the `VIARULE`, and a cut layer that is between the two routing layers.

`VIARULE viaRuleName` Specifies the name to identify the via rule.

`WIDTH minWidth TO maxWidth`
Specifies a wire width range. If the widths of two intersecting special wires fall within the wire width range, the `VIARULE` is used. To fall within the range, the widths must be greater than or equal to *minWidth* and less than or equal to *maxWidth*.

Note: `WIDTH` is defined by wire direction, not by layer. If you specify a `WIDTH` range, the rule applies to wires of the specified `DIRECTION` that fall within the range.

For example, if you specified the following `VIARULE`:

```
VIARULE viaRule1
    LAYER m1 ;
    WIDTH 0.5 TO 1.0 ;
    LAYER m2 ;
    WIDTH 1.0 TO 2.0 ;
    VIA via12_1 ;
    VIA via12_2 ;
END viaRule1
```

Whenever an *m1* wire with a width between 0.5 and 1.0 intersects with an *m2* wire with a width between 1.0 and 2.0, the via generation code attempts to put a `via12_1` at the intersection first. If the `via12_1` cause a DRC violation, a `via12_2` is then tried. If both fail, the default behavior from a `VIARULE GENERATE` statement for *m1* and *m2* is used.

LEF/DEF 5.4 Language Reference

LEF Syntax

Creating a Via from a Via Rule

You can specify how to create a via from a particular via rule in the following ways:

- List the vias explicitly, as described in this section. The router places the first via that does not create any design rule violations.
- Provide a via generation specification for creating the via. This method is shown in [“Via Rule Generate”](#) on page 74.

For Silicon Ensemble, when the `SROUTE`, `ADD WIRE`, and `MOVE WIRE` commands need a via, they examine the via rules *in the order of specification* in the LEF file for applicability. The first two layer and direction declarations must match the layers and directions of the intersecting wire segments.

If the widths of the wires fall in the ranges given by the `WIDTH` specifications, that via rule is used. If no via rule applies, the default via rule is used. To create a default via rule that covers all vias other than those whose widths are explicitly defined, omit the `WIDTH` statement when you create the via rule. To avoid `VERIFY LIBRARY` violations for undefined vias, a default via rule should exist that can be used for any wiring width not explicitly defined.

Changing Net Representation

For Silicon Ensemble, if you want to change the net representation from special to regular so that the automatic routers route the net, use `CHANGE NET REPRESENTATION` and do the following:

- Ensure that all wires in the net are the default or nondefault width for regular wiring. If not, the conversion does not occur.
- Ensure that the vias in the net are defined in the `VIA` section (defines vias for the default regular wiring) and the `NONDEFAULT RULE` section (defines vias for all regular wiring except the default wiring). If they are not, the vias will be missing after the conversion.

To change regular wiring to special wiring so that you can interactively route the net, use `CHANGE NET REPRESENTATION` and do the following:

- Ensure that all wire widths in the net are special wire widths.
- Ensure that all vias are defined in the `VIA RULE` section (defines vias for special wiring) or the `VIA RULE GENERATE` section.

LEF/DEF 5.4 Language Reference

LEF Syntax

Via Rule Generate

```
VIARULE viaRuleName GENERATE
    LAYER routingLayerName ;
        DIRECTION {HORIZONTAL | VERTICAL} ;
        [WIDTH minWidth TO maxWidth ;]
        [OVERHANG overhang ;]
        [METALOVERHANG metalOverhang ;]
    LAYER routingLayerName ;
        DIRECTION {HORIZONTAL | VERTICAL} ;
        [WIDTH minWidth TO maxWidth ;]
        [OVERHANG overhang ;]
        [METALOVERHANG metalOverhang ;]
    LAYER cutLayerName ;
        RECT pt pt ;
        SPACING xSpacing BY ySpacing ;
        [RESISTANCE resistancePerCut ;]

END viaRuleName
```

Defines formulas for generating wire arrays. Use the VIARULE GENERATE section to cover special wiring that is not explicitly defined in the VIARULE section.

Rather than specifying a list of vias for the situation, you can create a formula to specify how to generate the cut layer geometries.

DIRECTION {HORIZONTAL | VERTICAL}

Specifies the wire direction. If you specify a WIDTH range, the rule applies to wires of the specified DIRECTION that fall within the range. Otherwise, the rule applies to all wires of the specified DIRECTION on the layer.

For an example of a VIARULE GENERATE statement that uses DIRECTION, see [“Via Rule Generate Direction Example”](#) on page 77.

Note: OVERHANG and METALOVERHANG are defined by wire direction, not by layer. For the via generated, both metal layers (for example, *metal1* and *metal2*) will have the same METALOVERHANG and OVERHANG values. It is not possible to have different overhangs for different routing layers. You must set the overhang in both directions to the distance needed by the layer that needs the largest overhang.

GENERATE Defines a formula for generating the appropriate via.

LAYER *cutLayerName* Specifies the cut layer for the generated via.

LEF/DEF 5.4 Language Reference

LEF Syntax

LAYER *routingLayerName*

Specifies the routing layers for the top and bottom of the via.

METALOVERHANG *metalOverhang*

Specifies the amount of metal overhang between the edge of the wire and the edge of the via for the specified wire *DIRECTION*, for both metal layers. (See [Figure 1-4](#) on page 76.)

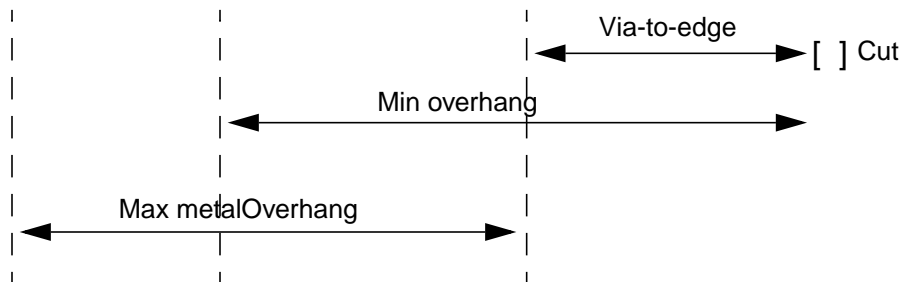
Default: The value that matches the overhang

Metal overhang is calculated in the following way:

$$(\text{width of via}) + (2 \times \text{overhang}) - (\text{width of wire}) / 2$$

Even though OVERHANG and METALOVERHANG define upper and lower bounds of a single property, their points of reference are different. (See [Figure 1-3](#) on page 75.)

Figure 1-3



OVERHANG = Min overhang

METALOVERHANG = Max metalOverhang

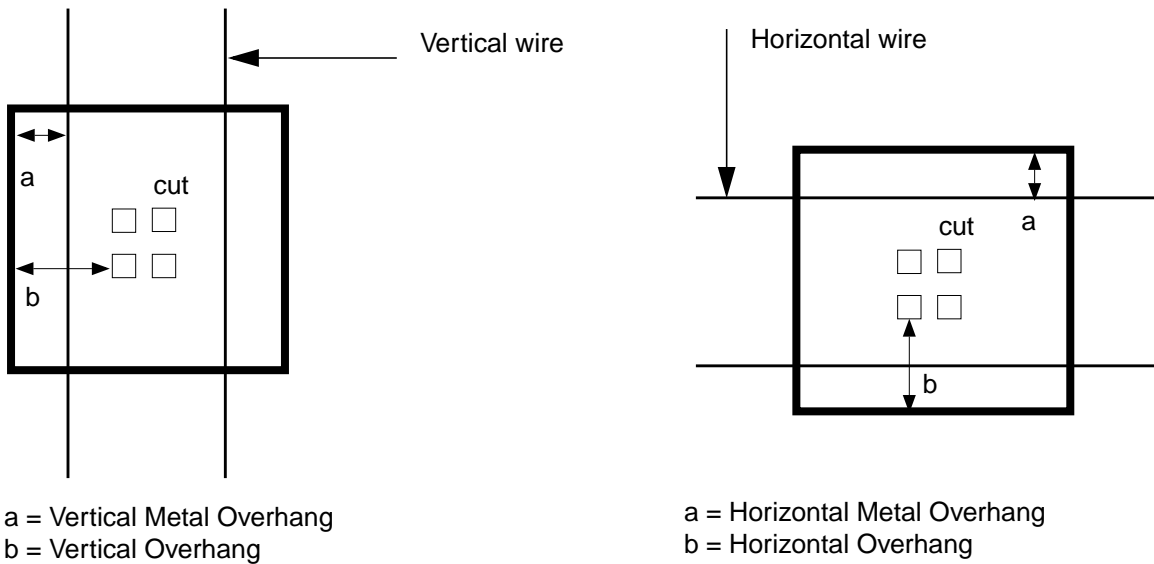
OVERHANG *overhang*

Specifies the amount of metal between contact cuts and the edge of the via for the specified wire *DIRECTION*, for both metal layers. (See [Figure 1-4](#) on page 76.)

LEF/DEF 5.4 Language Reference

LEF Syntax

Figure 1-4



`RECT pt pt` Specifies the location of the lower left contact cut rectangle.

`RESISTANCE resistancePerCut`
Specifies the resistance of the cut layer, given as the resistance per contact cut.
Type: Float

`SPACING xSpacing BY ySpacing`
Defines center-to-center spacing in the x and y dimensions to create an array of contact cuts. The number of cuts of an array in each direction is the most that can fit within the bounds of the intersection formed by the two special wires. Cuts are only generated where they do not violate stacked or adjacent via design rules.

`VIARULE viaRuleName` Specifies the name for the formula.

`WIDTH minWidth TO maxWidth`
Specifies a wire width range within which the wire must fall in order for the rule to apply. That is, the wire width must be greater than or equal to *minWidth* and less than or equal to *maxWidth*.

LEF/DEF 5.4 Language Reference

LEF Syntax

Note: WIDTH is defined by wire direction, not by layer. If you specify a WIDTH range, the rule applies to wires of the specified DIRECTION that fall within the range.

Turn Via Rules Information

Turn via rules fill in the corner area at the intersection between two special wires on the same layer, where each wire must be extended by half the width of the other wire. Single-layer turns in special wiring are defined as single-layer vias. Each routing layer needs to specify a turn rule with no size restriction and no rectangle.

Via Rule Generate Direction Example

The following example describes a formula for generating via cuts using DIRECTION. If you specify DIRECTION, the WIDTH range is based on the connecting horizontal or vertical wire, *not* the layer of the connecting wire. Therefore, a horizontal wire that is between 0 and 1.0 μm wide and a vertical wire that is between 0 and 2.0 μm wide will trigger ViaGen1, independent of whether the horizontal wire is *M1* or *M2*.

```
VIARULE viaGen1 GENERATE
  LAYER M1 ;
    DIRECTION HORIZONTAL ;
    WIDTH 0 TO 1.0 ;
    OVERHANG 0.5 ;
  LAYER M2 ;
    DIRECTION VERTICAL ;
    WIDTH 0 TO 2.0 ;
    OVERHANG 0.5 ;
  LAYER CUT12 ;
    ...
END viaGen1
```

LEF/DEF 5.4 Language Reference

LEF Syntax

DEFINE and ALIAS

This chapter contains information about the following topics.

- **DEFINE statements:**
 - ❑ [DEFINE Expressions](#) on page 80
 - ❑ [DEFINE Syntax](#) on page 83
 - ❑ [DEFINE Expansion](#) on page 84
 - ❑ [DEFINE Examples](#) on page 85
- **ALIAS statements:**
 - ❑ [ALIAS Definition](#) on page 86
 - ❑ [ALIAS Examples](#) on page 86
 - ❑ [ALIAS Expansion](#) on page 87

DEFINE Statements

DEFINE statements assign variable names to Boolean, numeric, and string expressions in a Library Exchange Format (LEF) or Design Exchange Format (DEF) file. **INPUT** LEF and **INPUT** DEF evaluate each expression when the LEF or DEF file is read. The variable remains fixed until redefined by a subsequent **DEFINE** statement. Once you create a **DEFINE** variable, it can subsequently appear in place of any LEF or DEF token. **DEFINE** statements are not saved into the database.

DEFINE variables are used in LEF and DEF files for substitution. For example, if you define the variables **&X** and **&Y**, you can use the following LEF statement:

```
SIZE &X BY &Y
```

Each expression has a fixed type, that is, numeric, string, or Boolean. The type is determined by the **DEFINE** token used: **&DEFINE** (numeric), **&DEFINES** (string), or **&DEFINEB** (Boolean).

LEF/DEF 5.4 Language Reference

DEFINE and ALIAS

The expression value and the `DEFINE` token must match. For example, when you use `&DEFINEB` to assign an expression, the expression must always evaluate to a Boolean value.

DEFINE Expressions

Expressions in `DEFINE` statements are written in either single- or multiple-line forms. Each single-line expression must appear entirely on one line of the LEF or DEF file. The three types of single-line expressions follow.

- `numericExpression`
- `booleanExpression`
- `stringExpresssion`

Each multiple-line expression has an IF/THEN/ELSE form and spans several lines. The three types of multiple-line expressions follow.

- `numericIfExpression`
- `booleanIfExpression`
- `stringIfExpression`

Note: Expressions can contain parentheses for grouping. However, you must insert a space or tab character between any parenthesis or operator token and an adjacent operand.

numericExpression

Valid numeric expressions evaluate to a numeric constant. You can use the following operands.

- Integer constant
- Real constant
- `numericExpressionName`

When an operand is a numeric expression name, you must have previously assigned a value using a `&DEFINE` statement. The value of *numericExpressionName* is used when evaluating the expression.

Numeric operators (in decreasing order of precedence) are as follows:

1. `()` (parentheses)
2. `–` (unary negation)

LEF/DEF 5.4 Language Reference

DEFINE and ALIAS

3. $\ast /$

4. $+ -$ (binary)

Otherwise, evaluation is from left to right. The following are examples of valid numeric expressions:

```
( 32234 + 31105 ) * 3 / 2
( &X1 - 22921 + &X2 / 2 ) * - 1.88E-2
&Z1 * ( ( &X2 - 3.3E2 ) / &Y3 )
```

stringExpression

Valid string expressions evaluate to a string constant and can have either of the following forms:

- `"stringText"`
- `stringExpressionName`

The *stringText* is any alphanumeric string and must be enclosed in quotes. The quote character (") is allowed within the string text because the string terminator is the quote character followed by a space, tab, or EOL. When a string expression is a *stringExpressionName*, it must have been previously assigned using a `&DEFINES` statement.

The following operators can be used in string expressions:

- `()` parenthesis
- `+` concatenation

For example:

```
&name + "-" + &type
```

booleanExpression

Valid Boolean expressions evaluate to the Boolean constants, true or false. Boolean expressions are generated by the following:

- Two numeric expressions connected by one of the following binary numeric operators:
 - EQ or = (equal) NEQ or <> (not equal)
 - LT or < (less than) GT or > (greater than)
 - LEQ or <= GEQ or >= (greater

LEF/DEF 5.4 Language Reference

DEFINE and ALIAS

(less than or equal)than or equal)

AND (non-zero = True) OR (non-zero = True)

- Two string expressions connected by one of the following binary string operators:

EQ or = (equal)NEQ or <> (not equal)

LT or < (less than)GT or > (greater than)

LEQ or <= GEQ or >=

For the above operators, the comparison looks at the character codes, starting from the left. Non-null strings are greater than null strings.

ANDOR

For AND/OR string expression evaluation, a non-null string is considered True, and a null string (" ") is considered False.

- Two Boolean expressions connected by one of the following binary Boolean operators:

AND (logical and)OR (logical or)

EQNEQ

- A Boolean, numeric, or string expression inverted by the unary Boolean operator NOT. (Numerics and strings evaluate to true or false.)
- *booleanExpressionName*
- Combinations of the above

When an operand is a Boolean expression name, it must have been previously assigned using a &DEFINEB statement. The value of *booleanExpressionName* is used when evaluating the expression. Operators used in Boolean expressions have the following decreasing order of precedence:

1. ()
2. – (unary)
3. * /
4. + – (binary)
5. EQ, NEQ, LT, GT, LEQ, GEQ, =, <>, <, >, <=, >=
6. NOT
7. AND
8. OR

LEF/DEF 5.4 Language Reference

DEFINE and ALIAS

Otherwise, evaluation is from left to right.

The following are examples of Boolean expressions:

```
&X1 > 22876
```

(If X1 is numeric.)

```
&FLAG <> "INT" AND &DELAY > 1.3E-3
```

(If &FLAG is string; &DELAY is numeric.)

```
NOT &T1 OR NOT &S1
```

(If &T1 and &S1 are Boolean.)

numericIFexpression

Numeric IF expressions have the following form:

```
IF { booleanExpression }  
  THEN { numericExpression | numericIFexpression }  
  ELSE { numericExpression | numericIFexpression }
```

stringIFexpression

String IF expressions have the following form:

```
IF { booleanExpression }  
  THEN { stringExpression | stringIFexpression }  
  ELSE { stringExpression | stringIFexpression }
```

booleanIFexpression

Boolean IF expressions have the following form:

```
IF { booleanExpression }  
  THEN { booleanExpression | booleanIFexpression }  
  ELSE { booleanExpression | booleanIFexpression }
```

DEFINE Syntax

DEFINE statements assign expressions to variables. Each type of expression (numeric, string, and Boolean) has a different DEFINE statement syntax, as shown below:

LEF/DEF 5.4 Language Reference

DEFINE and ALIAS

```
&DEFINE  numericExpressionName  =  
    { numericExpression  
    | numericIFexpression } ;  
&DEFINES stringExpressionName  =  
    { stringExpression  
    | stringIFexpression } ;  
&DEFINEB booleanExpressionName  =  
    { booleanExpression  
    | booleanIFexpression } ;
```

Variable names have the following restrictions:

- Must begin with an ampersand (&)
- Are case sensitive based on the value of `NAMESCASESENSITIVE` in the LEF input, or the value of `Input.Lef.Names.Case.Sensitive`
- Cannot contain spaces, tabs, or control characters

DEFINE statements have the following restrictions:

- `&DEFINE`, `&DEFINES`, and `&DEFINEB` are reserved keywords and must be the first token in any line in which they appear.
- `IF`, `THEN`, `ELSE`, `TRUE`, `FALSE`, and all operators are reserved tokens within a `DEFINE` statement.
- Spaces or tabs must be placed between all operators and operands, and between parentheses and operands.
- `DEFINE` statements end with a semicolon (;) as a delimiter.
- When a `DEFINE` statement assigns an expression name to a *numericExpression*, *stringExpression*, or *booleanExpression*, the definition must appear on the same line. A `DEFINE` statement can span multiple lines only if it assigns an expression name to an `IF` expression. In this case, the first `IF` keyword must appear on the same line as the `DEFINE` statement.
- When an expression name appears inside a `DEFINE` statement, the current value of the expression name is used to evaluate the `DEFINE` expression.

DEFINE Expansion

You can specify `DEFINE` symbols as expression expansion parameters for LEF or DEF files. A `DEFINE` can serve as a substitute for any token of a LEF or DEF file. `DEFINE` expansion can be considered as the reverse operation of expression definition. That is, to retrieve the

LEF/DEF 5.4 Language Reference

DEFINE and ALIAS

current *ExpressionName*, use *&&ExpressionName*. If an *ExpressionName* does not have a current value, no substitution occurs. All EOL (end of line), space, and tab characters are preserved in a string `DEFINE` statement.

Syntax for *ExpressionName* expansion is defined as follows:

&ExpressionName

where *ExpressionName* is any *Name* previously defined.

DEFINE Examples

The following are examples of numeric `DEFINE` statements:

```
&DEFINE &A1 = 2.5E4 ;
&DEFINE &RES = IF &MARK = "FREE"
                THEN &A1 + &B1
                ELSE &A1 - &B1 ;
```

The following are examples of string `DEFINE` statements:

```
&DEFINES &VERSION = "3.1.8" ;
&DEFINES &DIR = IF &TAG
                THEN "INPUT"
                ELSE "OUTPUT" ;
```

The following are examples of Boolean `DEFINE` statements:

```
&DEFINEB &TAG = &DONE AND NOT &SEL ;
&DEFINEB &FLAG = IF &X < &Y
                THEN TRUE
                ELSE FALSE ;
```

IF expressions can be nested as shown in the following examples:

```
&DEFINES&T1 = IF &X < &Y
                THEN IF &X > &Z
                        THEN "OK"
                        ELSE "BELOW"
                ELSE "ABOVE" ;

&DEFINE &P1 = IF &X < &Y
                THEN 3331 + &Y
                ELSE IF &Z < &X
                        THEN 3331 + &X
                        ELSE 3331 + &Z ;
```

ALIAS Statements

You can use alias statements in LEF and DEF files to define commands or parameters associated with the library or design. An alias statement can appear anywhere in a LEF or DEF file as follows:

```
&ALIAS  &&aliasName    =    aliasDefinition    &ENDALIAS
```

&ALIAS and &ENDALIAS are both reserved keywords and are not case sensitive. An alias statement has the following requirements:

- &ALIAS must be the first token in the line in which it appears.
- *aliasName* is string name and must appear on the same line as &ALIAS. It is case sensitive based on the value of NAMESCASESENSITIVE in the LEF input, or the value of `Input.Lef.Names.Case.Sensitive`.
- *aliasName* cannot contain any of the following special characters: #, space, tab, or control characters.
- &ENDALIAS must be the last token in the line in which it appears.
- Multiple commands can appear in the alias definition, separated by semicolons. However, the last command must not be terminated by a semicolon.

ALIAS Definition

The alias name (*aliasName*) is an identifier for the associated alias definition (*aliasDefinition*). The data reader stores the alias definition in the database. If the associated alias name already exists in the database, a warning is issued and the existing definition is replaced.

Alias definitions are text strings with the following properties:

- *aliasDefinition* is any text excluding "&ENDALIAS".
- All EOL, space, and tab characters are preserved.
- *aliasDefinition* text can expand to multiple lines.

ALIAS Examples

The following examples include legal and illegal alias statements:

- The following statement is legal.

LEF/DEF 5.4 Language Reference

DEFINE and ALIAS

```
&ALIAS &&MAC = SROUTE ADDCELL AREA &&CORE &ENDALIAS
```

- The following statement is illegal because MAC does not start with “&&”.

```
&ALIAS MAC = SROUTE AREA &&CORE &ENDALIAS
```

- The following statement is illegal because &ALIAS is not the first token in this line.

```
( 100 200 ) &ALIAS &&MAC = SROUTE AREA &&CORE &ENDALIAS
```

- The following statement is legal. It contains multiple commands; the last command is not terminated by a semicolon.

```
$ALIAS $$ = INPUT LEF myfile.txt;  
VERIFY LIBRARY  
ENDALIAS
```

The following examples show legal and illegal alias names:

- “Engineer_change” is a legal alias name.

```
&&Engineer_change
```

- “&Version&History&&” is a legal alias name.

```
&&&Version&History&&
```

- “design history” is an illegal alias name. It contains a space character and is considered as two tokens: an *aliasName* token “&&design,” and a non-*aliasName* token “history”.

```
&&design history
```

- “someName#IO-pin-Num” is an illegal alias name. It contains a “#” character and is translated as one *aliasName* token “&&someName”. The “#” is considered a comment character.

```
&&someName#IO-pin-Num
```

ALIAS Expansion

Alias expansion is the reverse operation of alias definition. The following is the syntax for alias expansion.

```
&&aliasName
```

where *aliasName* is any name previously defined by an alias statement. If an *aliasName* does not exist in the database, no substitution occurs.

You use aliases as string expansion parameters for LEF or DEF files. An alias can substitute for any token of a LEF or DEF file.

LEF/DEF 5.4 Language Reference

DEFINE and ALIAS

Working with LEF

This chapter contains information about the following topics.

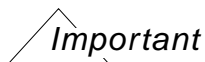
- [Incremental LEF](#) on page 89
- [Error Checking](#) on page 90

Incremental LEF

`INPUT LEF` can add new data to the current database, providing an incremental LEF capability. Although it is possible to put an entire LEF library in one file, some systems require that you put certain data in separate files.

This feature also is useful, for example, when combined with the `INPUT GDSII` command, to extract geometric data from a GDSII-format file and add the data to the database.

When using `INPUT LEF` on a database that has been modified previously, save the previous version before invoking `INPUT LEF`. This provides a backup in case the library information has problems and the database gets corrupted or lost.



The original LEF file, created with `FINPUT LEF` (or with `INPUT LEF` when no database is loaded), must contain all the layers.

Adding Objects to the Library

`INPUT LEF` can add the following objects to the database:

- New via
- New via rule
- Samenet spacings (if none have been specified previously)
- New macro

LEF/DEF 5.4 Language Reference

Working with LEF

If geometries have not been specified for an existing via, `INPUT LEF` can add layers and associated rectangle geometries. If not specified previously for a macro, `INPUT LEF` can add the following:

- `FOREIGN` statement
- `EEQ`
- `LEQ`
- `Size`
- `Overlap geometries`
- `Obstruction geometries`

If not previously specified for an existing macro pin, `INPUT LEF` can add the following:

- `Mustjoins`
- `Ports and geometries`

The database created by `INPUT LEF` can contain a partial library. Run `VERIFY LIBRARY` before proceeding.

If new geometries are added to a routed database, run `VERIFY GEOMETRY` and `VERIFY CONNECTIVITY` to identify new violations.

Important

When defining a pin with no port geometries with the intent of incrementally adding them, *do not* include an empty `PORT` statement as shown below.

```
MACRO abc
...
  PIN a
    ...
    PORT # dummy pin-port, do not
    END  # include these two lines
  END a
  ...
```

Error Checking

To help develop, test, and debug generic libraries and parametric macros, LEF and DEF have a user-defined error checking facility. This facility consists of seven utilities that you can use

LEF/DEF 5.4 Language Reference

Working with LEF

from within a LEF or DEF file during the scanning phase of LEF/DEF readers. These utilities have the following features:

- A message facility that writes to one or more text files during LEF or DEF input
- An error handling facility that logs user detected warnings, errors, and fatal errors

The error checking utilities have the following syntax:

```
&CREATEFILE &fileAlias =  
    { stringExpression  
      | stringIF-ELSEexpression } ;  
&OPENFILE &fileAlias ;  
&CLOSEFILE &fileAlias ;  
&MESSAGE  
    {&fileAlias | &MSGWINDOW} = message ;  
&WARNING  
    {&fileAlias | &MSGWINDOW} = message ;  
&ERROR  
    {&fileAlias | &MSGWINDOW} = message ;  
&FATALERROR  
    {&fileAlias | &MSGWINDOW} = message ;  
message =  
    { &fileAlias | stringExpression  
      | stringIF-ELSEexpression  
      | stringIFexpression }
```

Message Facility

The message facility outputs user-defined messages during the scanning phase of LEF and DEF input. These messages can be directed to the message window.

&CREATEFILE

The &CREATEFILE utility first assigns a token (*&fileAlias*) to represent a named file. The file name is derived from a previously defined string, a quoted string, or an IF-ELSE expression that evaluates to a string. The following example illustrates these three cases.

```
&DEFINES &messagefile = "demo1.messages" ;  
&CREATEFILE &outfile = &messagefile ;  
&CREATEFILE &msgs =  
    "/usr/asics/cmos/fif4/errors.txt" ;
```

LEF/DEF 5.4 Language Reference

Working with LEF

```
&CREATEFILE &messages =  
    IF &errortrap  
        THEN "errs.txt"  
        ELSE "/dev/null" ;
```

The derived file name must be a legal file name in the host environment. The default directory is the current working directory. The file names are case sensitive.

`&CREATEFILE` creates an empty file with the given name and opens the file. If the token is already bound to another open file, a warning is issued, the file is closed, and the new file is opened. If the file already exists, the version number is incremented.

&CLOSEFILE and &OPENFILE

The `&CLOSEFILE` utility closes the file bound to a given token; `&OPENFILE` opens the file bound to a given token. `&CLOSEFILE` and `&OPENFILE` control the number of open files. Each operating system has a limit for the number of open files. Therefore, `&CLOSEFILE` might be needed to free up extra file descriptors.

Files are closed in the following ways.

- All user files are closed at the end of the scanning phase of the LEF and DEF readers.
- All user files are closed if the scanning phase aborts.
- If `&CREATEFILE` is invoked with a token that is already bound to an open file, that file is closed before opening the new file.

&MESSAGE

The `&MESSAGE` utility appends text to the file represented by the *&fileAlias* token, or to the message window if `&MSGWINDOW` is specified.

`&MSGWINDOW` is a special file alias that is not created, opened, or closed. The assigned expression (right side of the statement) can be one of the following:

<i>&fileAlias</i>	Must correspond to a valid file that has been successfully opened. The contents of the file are appended to the target file (or message window).
-----------------------	--

stringExpression

Either a string or a string token.

For example:

LEF/DEF 5.4 Language Reference

Working with LEF

```
&DEFINES &romword16 =  
    "ROM word size = 16 bits" ;  
&MESSAGE &msgs = "ROM size = 256" ;  
&MESSAGE &msgs = &romword16 ;
```

stringIF-ELSEexpression

String IF-ELSE expressions evaluate a Boolean expression and then branch to string values, for example:

```
&&MESSAGE &msgs =  
    IF (&&c_flag = 0)  
        THEN "FLAG C set to 0"  
    ELSE IF ( &&c_flag = 1 )  
        THEN "FLAG C set to 1"  
    ELSE "FLAG C set to 2" ;
```

As shown in this example, IF-ELSE expressions can be nested.

stringIFexpression

A string IF expression is an IF-ELSE expression without the ELSE phrase. The Boolean expression is evaluated, and if true, the THEN string is sent to the target file; if false, no string is sent, for example,

```
&MESSAGE &msgs =  
    IF ( &&buf = "INV_BIG" )  
        THEN "INV_BIG buffers" ;
```

Neither the file alias token nor &MSGWINDOW can be part of the assigned expression.

Error-Checking Facility

In addition to the message facility, you have partial control of the error checking facility of the LEF and DEF readers. When scanning LEF or DEF input, the readers record warnings, errors, and fatal errors. At the end of the scan, the total number of each is sent to the message window before proceeding with the reader phase.

If a fatal error is detected, input is aborted after the scanning phase.

With the user interface to the error checking facility, the LEF and DEF files can include custom error checking. User detected warnings, errors, and fatal errors, can be logged, thereby incrementing the DEF/LEF reader's warning, error, and fatal error counts.

LEF/DEF 5.4 Language Reference

Working with LEF

A user-detected fatal error terminates input just as with the resident error checking facility. In addition, the user defined error checking facility utilities can send message strings to the message window.

&WARNING, &ERROR, and &FATALERROR

The `&WARNING`, `&ERROR`, and `&FATALERROR` utilities use the same syntax as the `&MESSAGE` utility. These utilities can send message strings to files and to the message window in the same manner as `&MESSAGE`. In addition, when the assigned expression is a string `IF` expression, or a string `IF-ELSE` expression, then the associated counter (warnings, errors, or fatal errors) is incremented by 1 if any `IF` condition evaluates to true.

DEF Syntax

This chapter contains information about the following topics:

■ [About Design Exchange Format Files](#) on page 96

- ❑ [General Rules](#) on page 96
- ❑ [Order of DEF Statements](#) on page 97

■ [DEF Statement Definitions](#) on page 98

- ❑ [Blockages](#) on page 98
- ❑ [Bus Bit Characters](#) on page 99
- ❑ [Components](#) on page 100
- ❑ [Design](#) on page 102
- ❑ [Die Area](#) on page 103
- ❑ [Divider Character](#) on page 103
- ❑ [Extensions](#) on page 103
- ❑ [Fills](#) on page 104
- ❑ [GCell Grid](#) on page 105
- ❑ [Groups](#) on page 106
- ❑ [History](#) on page 108
- ❑ [Names Case Sensitive](#) on page 108
- ❑ [Nets](#) on page 108
- ❑ [Regular Wiring Statement](#) on page 114
- ❑ [Pins](#) on page 117
- ❑ [Pin Properties](#) on page 124

LEF/DEF 5.4 Language Reference

DEF Syntax

- ❑ [Property Definitions](#) on page 125
- ❑ [Regions](#) on page 128
- ❑ [Rows](#) on page 129
- ❑ [Scan Chains](#) on page 130
- ❑ [Slots](#) on page 134
- ❑ [Special Nets](#) on page 135
- ❑ [Special Wiring Statement](#) on page 137
- ❑ [Technology](#) on page 142
- ❑ [Tracks](#) on page 142
- ❑ [Units](#) on page 143
- ❑ [Version](#) on page 144
- ❑ [Vias](#) on page 145

About Design Exchange Format Files

A Design Exchange Format (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 an ASCII representation using the syntax conventions described in [“Typographic and Syntax Conventions”](#) on page 10.

DEF conveys logical design data to, and physical design data from, place-and-route tools. Logical design data can include internal connectivity (represented by a netlist), grouping information, and physical constraints. Physical data includes placement locations and orientations, routing geometry data, and logical design changes for backannotation. Place-and-route tools also can read physical design data, for example, to perform ECO changes.

For standard-cell-based/ASIC flow tools, floorplanning is part of the design flow. You typically use the various floorplanning commands to interactively create a floorplan. This data then becomes part of the physical data output for the design using the ROWS, TRACKS, GCELLGRID, and DIEAREA statements. You also can manually enter this data into DEF to create the floorplan.

General Rules

Note the following information about creating DEF files:

LEF/DEF 5.4 Language Reference

DEF Syntax

- Lines in the DEF file are limited to 2,048 characters (extra characters are truncated on input).
- Net names and cell names also are limited to 2,048 characters.
- DEF statements end with a semicolon (;). You *must* leave a space before the semicolon.
- Each section can be specified only once. Sections end with `END SECTION`.
- You must define all objects before you reference them except for the + ORIGINAL argument in the NETS section.
- No regular expressions or wildcard characters are recognized except for the (* *pinName*) argument in the SPECIALNETS section.

Order of DEF Statements

Standard DEF files can contain the following statements and sections. You must specify the statements and sections in the following order.

```
VERSION statement
NAMECASESENSITIVE statement
DIVIDERCHAR statement
BUSBITCHARS statement
DESIGN statement
[ TECHNOLOGY statement ]
[ UNITS statement ]
[ HISTORY statement ]...
[ PROPERTYDEFINITIONS section ]
[ DIEAREA statement ]
[ ROWS statement ] ...
[ TRACKS statement ]...
[ GCELLGRID statement ]...
[ VIAS statement ]
[ REGIONS statement ]
COMPONENTS section
[ PINS section ]
[ PINPROPERTIES section ]
[ BLOCKAGES section ]
[ SLOTS section ]
[ FILLS section ]
[ SPECIALNETS section ]
NETS section
[ SCANCHAINS section ]
[ GROUPS section ]
[ BEGINEXT section ]
END DESIGN statement
```

LEF/DEF 5.4 Language Reference

DEF Syntax

Note: You can include `BEGINEXT` sections at any point.

DEF Statement Definitions

The following definitions describe the syntax arguments for the statements and sections that make up a DEF file. The statements and sections are listed in alphabetical order, *not* in the order they must appear in a DEF file. For the correct order, see [“Order of DEF Statements”](#) on page 97.

Blockages

```
[BLOCKAGES numBlockages ;
  [- LAYER layerName
    [+ COMPONENT compName | + SLOTS | + FILLS | + PUSHDOWN]
    {RECT pt pt} ...
  ;] ...
  [- PLACEMENT
    [+ COMPONENT compName | + PUSHDOWN]
    {RECT pt pt} ...
  ;] ...
END BLOCKAGES]
```

Defines placement and routing blockages in the design. You can define simple blockages (blockages specified for an area), or blockages that are associated with specific instances (components). You can only associate blockages with placed instances. If you move the instance, its blockage moves with it.

<code>COMPONENT compName</code>	Specifies a component with which to associate a blockage. Specify with <code>LAYER layerName</code> to create a routing blockage associated with a component. Specify with <code>PLACEMENT</code> to create a placement blockage associated with a component.
<code>FILLS</code>	Creates a blockage on the specified layer where metal fills cannot be placed.
<code>LAYER layerName</code>	Specifies the layer on which to create a blockage.
<code>numBlockages</code>	Specifies the number of blockages in the design.
<code>PLACEMENT</code>	Creates a placement blockage. You can create a simple placement blockage, or a placement blockage attached to a specific component.

LEF/DEF 5.4 Language Reference

DEF Syntax

PUSHDOWN	Specifies that the blockage was pushed down into the block from the top level of the design.
RECT <i>pt pt</i>	Specifies the coordinates of the blockage geometry. The coordinates you specify are absolute. If you associate a blockage with a component, the coordinates are not relative to the component's origin.
SLOTS	Creates a blockage on the specified layer where slots cannot be placed.

Example

The following example shows a BLOCKAGES section that defines eight blockages in the following order: two *metal2* routing blockages, a pushed down routing blockage, a routing blockage attached to component |i4, a floating placement blockage, a pushed down placement blockage, a placement blockage attached to component |i3, and a fill blockage.

```
BLOCKAGES 7 ;
- LAYER metall
  RECT ( -300 -310 ) ( 320 330 )
  RECT ( -150 -160 ) ( 170 180 ) ;
- LAYER metall + PUSHDOWN
  RECT ( -150 -160 ) ( 170 180 ) ;
- LAYER metall + COMPONENT |i4
  RECT ( -150 -160 ) ( 170 180 ) ;
- PLACEMENT
  RECT ( -150 -160 ) ( 170 180 ) ;
- PLACEMENT + PUSHDOWN
  RECT ( -150 -160 ) ( 170 180 ) ;
- PLACEMENT + COMPONENT |i3
  RECT ( -150 -160 ) ( 170 180 ) ;
- LAYER metall + FILLS
  RECT ( -160 -170 ) ( 180 190 ) ;

END BLOCKAGES
```

Bus Bit Characters

```
BUSBITCHARS "delimiterPair" ;
```

Specifies the pair of characters used to specify bus bits when DEF names are mapped to or from other databases. The characters must be enclosed in double quotation marks. For example:

```
BUSBITCHARS "()" ;
```

LEF/DEF 5.4 Language Reference

DEF Syntax

If one of the bus bit characters appears in a DEF name as a regular character, you must use a backslash (\) before the character to prevent the DEF reader from interpreting the character as a bus bit delimiter.

Components

```
COMPONENTS numComps ;
  [- compName modelName
    [+ EEQMASTER macroName]
    [+ SOURCE {NETLIST | DIST | USER | TIMING}]
    [+ FOREIGN foreignCellName pt orient]
    [+ {FIXED pt orient | COVER pt orient | PLACED pt orient
        | UNPLACED} ]
    [+ WEIGHT weight]
    [+ REGION {pt pt | regionName}]
    [+ PROPERTY {propName propVal} ...]...
  ;] ...

END COMPONENTS
```

Defines design components, their location, and associated attributes.

compName modelName Specifies the component name in the design, which is an instance of *modelName*, the name of a model defined in the library. A *modelName* must be specified with each *compName*.

COVER pt orient Specifies that the component has a location and is a part of a cover macro. A *COVER* component cannot be moved by automatic tools or interactive commands. You must specify the component's location and its orientation.

EEQMASTER macroName Specifies that the component being defined should be electrically equivalent to the previously defined *macroName*.

FIXED pt orient Specifies that the component has a location and cannot be moved by automatic tools, but can be moved using interactive commands. You must specify the component's location and orientation.

FOREIGN foreignCellName pt orient
The *FOREIGN* parameter is available if you want to interface to a foreign format such as GDSII. In LEF, a macro can be defined with *FOREIGN* cell references. The *FOREIGN* parameter in the DEF repeats the LEF specification for the component. The

LEF/DEF 5.4 Language Reference

DEF Syntax

FOREIGN parameter is ignored by DEF readers that use LEF for macro definitions.

Specifies the foreign (GDSII) structure name to use when placing an instance of the component. You must also specify the component origin (lower left corner when the component is in north orientation) offset from the foreign origin, and the orientation of the foreign cell when the component is in north orientation.

numComps Specifies the number of components defined in the COMPONENTS section.

PLACED *pt orient* Specifies that the component has a location, but can be moved using automatic layout tools. You must specify the component's location and orientation.

PROPERTY *propName propVal* Specifies a numerical or string value for a component property defined in the PROPERTYDEFINITIONS statement. The *propName* you specify must match the *propName* listed in the PROPERTYDEFINITIONS statement.

REGION {*pt pt* | *regionName*} Specifies a region in which the component must lie. *regionName* specifies a region already defined in the REGIONS section. If the region is smaller than the bounding rectangle of the component itself, the DEF reader issues an error message and ignores the argument. If the region does not contain a legal location for the component, the component remains unplaced after the placement step.

SOURCE {NETLIST | DIST | USER | TIMING} Specifies the source of the component.
Value: Specify one of the following:

DIST	Component is generated using clock tree or net synthesis tools.
NETLIST	Component is specified in the netlist.
TIMING	Component is logical rather than physical change to the netlist, as with CTGEN or placement based synthesis.

LEF/DEF 5.4 Language Reference

DEF Syntax

USER Component is generated manually.









For Silicon Ensemble, the `SOURCE` field is output by the DEF writer to indicate where the component was created: netlist input, distribution input (as with `SROUTE ADDCELL`), or user input (as with `MOVE CELL NEW`). Components with `SOURCE TIMING` represent logical rather than physical changes to the netlist (as with buffer insertion by `CTGEN` or placement-based synthesis). The value of this field is preserved when input to the DEF reader.

UNPLACED Specifies that the component does not have a location.

WEIGHT *weight* Specifies the weight of the component., which determines whether or not automatic placement attempts to keep the component near the specified location. *weight* is only meaningful when the component is placed. All non-zero weights have the same effect during automatic placement.
Default: 0

Orientation Information

If a component has a location, you must specify its location and orientation. A component can have any of the following orientations:

Value	Definition	Value	Definition
N	North 	FN	Flipped north 
S	South 	FS	Flipped south 
W	West 	FW	Flipped west 
E	East 	FE	Flipped east 

Components are always placed such that the lower left corner of the cell is the origin (0,0) after any orientation. When a component flips about the y axis, it flips about the component center. When a component rotates, the lower left corner of the bounding box of the component's sites remains at the same placement location.

Design

DESIGN *designName* ;

LEF/DEF 5.4 Language Reference

DEF Syntax

Specifies a name for the design. The DEF reader reports a warning if this name is different from that in the database. In case of a conflict, the just specified name overrides the old name.

Die Area

```
[DIEAREA pt pt ;]
```

Specifies the corner points of the bounding rectangle for the design. All objects *must* be defined within this rectangle. *pt* is a float number.

Divider Character

```
DIVIDERCHAR "character" ;
```

Specifies the character used to express hierarchy when DEF names are mapped to or from other databases. The character must be enclosed in double quotation marks. For example:

```
DIVIDERCHAR "/" ;
```

If the divider character appears in a DEF name as a regular character, you must use a backslash (\) before the character to prevent the DEF reader from interpreting the character as a hierarchy delimiter.

Extensions

```
[BEGINEXT "tag"  
    extensionText
```

```
ENDEXT]
```

Adds customized syntax to the DEF file that can be ignored by tools that do not use that syntax. You can also use extensions to add new syntax not yet supported by your version of LEF/DEF, if you are using version 5.1 or later. You can add extensions as separate sections, or within other sections.

extensionText Defines the contents of the extension.

"*tag*" Identifies the extension block. You must enclose *tag* in quotes.

LEF/DEF 5.4 Language Reference

DEF Syntax

Examples

```
BEGINEXT "lVSI Signature 1.0"
    CREATOR "company name"
    DATE "timestamp"
    REVISION "revision number"
ENDEXT
```

Fills

```
[FILLS numFills ;
    [- LAYER layerName
        {RECT pt pt} ...
    ;] ...
END FILLS]
```

Defines the rectangular shapes that represent metal fills in the design. Each fill is defined as an individual rectangle.

LAYER layerName Specifies the layer on which to create the fill.

numFills Specifies the number of *LAYER* statements in the *FILLS* statement, *not* the number of rectangles.

RECT pt pt Specifies the lower left and upper right corner coordinates of the fill geometry.

Examples

The following example defines fills for layers MET1 and MET2.

```
FILLS 2 ;
- LAYER MET1
    RECT (1000 2000) (1500 4000)
    RECT (2000 2000) (2500 4000)
    RECT (3000 2000) (3500 4000) ;
- LAYER MET2
    RECT (1000 2000) (1500 4000)
    RECT (1000 4500) (1500 6500)
    RECT (1000 7000) (1500 9000)
    RECT (1000 9500) (1500 11500) ;
END FILLS
```


LEF/DEF 5.4 Language Reference

DEF Syntax

GCell Grid

```
[GCELLGRID
    {X start DO numColumns+1 STEP space} ...
    {Y start DO numRows+1 STEP space ;} ...]
```

Defines the gcell grid for a standard cell-based design. Each `GCELLGRID` statement specifies a set of vertical (X) and horizontal (Y) lines, or tracks, that define the gcell grid.

<code>DO numColumns+1</code>	Specifies the number of columns in the grid.
<code>DO numRows+1</code>	Specifies the number of rows in the grid.
<code>STEP space</code>	Specifies the spacing between tracks.
<code>X start, Y start</code>	Specify the location of the first vertical (x) and first horizontal (y) track.

GCell Grid Boundary Information

The boundary of the gcell grid is the rectangle formed by the extreme vertical and horizontal lines. The gcell grid partitions the routing portion of the design into rectangles, called gcells. The lower left corner of a gcell is the origin. The X size of a gcell is the distance between the upper and lower bounding vertical lines, and the Y size is the distance between the upper and lower bounding horizontal lines.

For example, the grid formed by the following two `GCELLGRID` statements creates gcells that are all the same size (100 x 200 in the following):

```
GCELLGRID X 1000 DO 101 STEP 100 ;
GCELLGRID Y 1000 DO 101 STEP 200 ;
```

A gcell grid in which all gcells are the same size is called a uniform gcell grid. Adding `GCELLGRID` statements can increase the granularity of the grid, and can also result in a nonuniform grid, in which gcells have different sizes.

For example, adding the following two statements to the above grid generates a nonuniform grid:

```
GCELLGRID X 3050 DO 61 STEP 100 ;
GCELLGRID Y 5100 DO 61 STEP 200 ;
```

When a track segment is contained inside a gcell, the track segment belongs to that gcell. If a track segment is aligned on the boundary of a gcell, that segment belongs to the gcell only if it is aligned on the left or bottom edges of the gcell. Track segments aligned on the top or right edges of a gcell belong to the next gcell.

LEF/DEF 5.4 Language Reference

DEF Syntax

GCell Grid Restrictions

Every track segment must belong to a gcell, so gcell grids have the following restrictions:

- The X coordinate of the last vertical track must be less than, and not equal to, the X coordinate of the last vertical gcell line.
- The Y coordinate of the last horizontal track must be less than, and not equal to, the Y coordinate of the last horizontal gcell line.

Gcells grids also have the following restrictions:

- Each GCELLGRID statement must define two lines.
- Every gcell need not contain the vertex of a track grid. But, those that do must be at least as large in both directions as the default wire widths on all layers.

Missing GCell Grid Statements

Default cases occur when the GCELLGRID statements for the X direction, Y direction, or both, are missing. Then INITIALIZE FLOORPLAN automatically creates a gcell grid in the missing direction that

- Starts at the first track and covers the last track.
- Uses a gcell X or Y size set to 15 via-line spacings.
- Uses the via-line spacing from the layer that has the preferred direction the same as the direction of the missing GCELLGRID statement. If two or more layers have the same preferred direction, the smallest via-line spacing is used.

You select a gcell grid based on routing resources for the floorplan. Available tracks can be actual channels such as core channels or the channeled areas around I/O pads. They also can be quasichannels created by the underlying row pattern or the layout of pin rails in the components. Blocked tracks are unavailable for routing in a given direction because of component obstructions or pins.

Groups

```
[GROUPS numGroups ;
  [- groupName compNameRegExpr ...
    [+ SOFT [MAXHALFPERIMETER value] [MAXX value] [MAXY value]]
    [+ REGION {pt pt | regionName} ]
    [+ PROPERTY {propName propVal} ...] ...
  ;] ...
```

LEF/DEF 5.4 Language Reference

DEF Syntax

END GROUPS]

Defines groups in a design.

compNameRegExpr Specifies the components that make up the group. Do not assign any component to more than one group. You can specify any of the following::

- A component name, for example C3205
- A list of component names separated by spaces, for example, I01 I02 C3204 C3205
- A regular expression for a set of components, for example, IO* and C320*

groupName Specifies the name for a group of components.

MAXHALFPERIMETER value Specifies the maximum half-perimeter for the smallest bounding rectangle containing all of the components in the group (after placement).

MAXX value Specifies the maximum x spacing between components in the group. To avoid problems finding an initial placement, MAXX should be at least 0.75 times MAXHALFPERIMETER.
Default: The value for MAXHALFPERIMETER, or the half-perimeter of the entire chip.

MAXY value Specifies the maximum y spacing between components in the group. To avoid problems finding an initial placement, MAXY should be at least 0.75 times MAXHALFPERIMETER.
Default: The value for MAXHALFPERIMETER, or the half-perimeter of the entire chip.

numGroups Specifies the number of groups defined in the GROUPS section.

PROPERTY propName propVal Specifies a numerical or string value for a group property defined in the PROPERTYDEFINITIONS statement. The *propName* you specify must match the *propName* listed in the PROPERTYDEFINITIONS statement.

REGION {pt pt | regionName} Specifies a rectangular region in which the group must lie. You

LEF/DEF 5.4 Language Reference

DEF Syntax

can specify the coordinates of the region, or specify the name of a region defined in the REGIONS section.

If you specify a region, MAXHALFPERIMETER, MAXX, and MAXY default to the entire chip. If region restrictions are specified in both COMPONENT and GROUP statements for the same component, the component restriction overrides the group restriction.

SOFT

Clusters the specified components during placement. The components in a soft group are not fixed in relative placement and orientation, but all components in a soft group are placed close together, based on the maximum half-perimeter specification.

History

```
[HISTORY anyText ;] ...
```

Lists a historical record about the design. Each line indicates one record. Any text excluding a semicolon (;) can be included in *anyText*. The semicolon terminates the HISTORY statement. Linefeed and Return do not terminate the HISTORY statement. Multiple HISTORY lines can appear in a file.

Names Case Sensitive

```
NAMESCASESENSITIVE {OFF | ON} ;
```

Determines if DEF data is case sensitive. The NAMESCASESENSITIVE statement must be on the first noncommented line after the VERSION statement. The case sensitivity of the DEF data must match that of the LEF data used in the design.

Nets

```
NETS numNets ;  
  [- { netName  
    [ ( {compName pinName | PIN pinName} [+ SYNTHESIZED] ) ] ...  
    | MUSTJOIN ( compName pinName ) }  
    [+ SHIELDNET shieldNetName ] ...  
    [+ VPIN vpinName [LAYER layerName] pt pt  
      [PLACED pt orient | FIXED pt orient | COVER pt orient] ] ...  
    [+ SUBNET subnetName  
      [ ( {compName pinName | PIN pinName | VPIN vpinName} ) ] ...  
      [NONDEFAULTRULE rulename]
```

LEF/DEF 5.4 Language Reference

DEF Syntax

```
    [regularWiring] ...] ...  
[+ XTALK class]  
[+ NONDEFAULTRULE ruleName]  
[regularWiring] ...  
[+ SOURCE {DIST | NETLIST | TEST | TIMING | USER}]  
[+ ORIGINAL netName]  
[+ USE {ANALOG | CLOCK | GROUND | POWER | RESET | SCAN | SIGNAL  
      | TIEOFF}]  
[+ PATTERN {BALANCED | STEINER | TRUNK | WIREDLOGIC}]  
[+ ESTCAP wireCapacitance]  
[+ WEIGHT weight]  
[+ PROPERTY {propName propVal} ...] ...  
;] ...  
  
END NETS
```

Defines netlist connectivity for nets containing regular pins. The default design rules apply to these pins, and the regular routers route to these pins. The `SPECIALNETS` statement defines netlist connectivity for nets containing special pins.

Input arguments for a net can appear in the `NETS` section or the `SPECIALNETS` section. In case of conflicting values, the DEF reader uses the last value encountered. `NETS` and `SPECIALNETS` statements can appear more than once in a DEF file. If a particular net has mixed wiring or pins, specify the special wiring and pins first.

You can also specify the netlist in the `COMPONENTS` statement. If the netlist is specified in both `NETS` and `COMPONENTS` statements, and if the specifications are not consistent, an error message appears. On output, the writer outputs the netlist in either format, depending on the arguments of the output command.

<code>compName pinName</code>	Specifies the name of a regular component pin on a net or a subnet. If a subnet includes regular pins, the regular pins must be included in the parent net.
-------------------------------	---

<code>COVER pt orient</code>	Specifies that the pin has a location and is a part of the cover macro. A <code>COVER</code> pin cannot be moved by automatic tools or by interactive commands. You must specify the pin's location and orientation.
------------------------------	--

<code>ESTCAP wireCapacitance</code>	Specifies the estimated wire capacitance for the net. <code>ESTCAP</code> can be loaded with simulation data to generate net constraints for timing-driven layout.
-------------------------------------	--

LEF/DEF 5.4 Language Reference

DEF Syntax

<code>FIXED <i>pt orient</i></code>	Specifies that the pin has a location and cannot be moved by automatic tools, but can be moved by interactive commands. You must specify the pin's location and orientation.
<code>LAYER <i>layerName</i></code>	Specifies the layer on which the virtual pin lies.
<code>MUSTJOIN (<i>compName pinName</i>)</code>	<p>Specifies that the net is a mustjoin. If a net is designated MUSTJOIN, its name is generated by the system. Only one net should connect to any set of mustjoin pins. Mustjoin pins for macros are defined in LEF. The only reason to specify a MUSTJOIN net in DEF (identified arbitrarily by one of its pins) is to specify prewiring for the MUSTJOIN connection.</p> <p>Otherwise, nets are generated automatically where needed for mustjoin connections specified in the library. If the input file specifies that a mustjoin pin is connected to a net, the DEF reader connects the set of mustjoin pins to the same net. If the input file does not specify connections to any of the mustjoin pins, the DEF reader creates a local MUSTJOIN net.</p>
<code><i>netName</i></code>	Specifies the name for the net. Each statement in the NETS section describes a single net. There are two ways of identifying the net: <i>netName</i> or MUSTJOIN. If the <i>netName</i> is given, a list of pins to connect to the net also can be specified. Each pin is identified by a component name and pin name pair (<i>compName pinName</i>) or as an I/O pin (<code>PIN <i>pinName</i></code>). Parentheses ensure readability of output. The keyword MUSTJOIN cannot be used as a <i>netName</i> .
<code>NONDEFAULTRULE <i>ruleName</i></code>	Specifies the LEF-defined nondefault rule to use when creating the net and wiring. When specified with SUBNET, names the nondefault rule to use when creating the subnet and its wiring.
<code><i>numNets</i></code>	Specifies the number of nets defined in the NETS section.
<code>ORIGINAL <i>netName</i></code>	Specifies the original net partitioned to create multiple nets, including the net being defined.
<code>PATTERN {BALANCED STEINER TRUNK WIREDLOGIC}</code>	Specifies the routing pattern used for the net.

LEF/DEF 5.4 Language Reference

DEF Syntax

Default: STEINER

Value: Specify one of the following:

BALANCED	Used to minimize skews in timing delays for clock nets.
STEINER	Used to minimize net length.
TRUNK	Used to minimize delay for global nets.
WIREDLOGIC	Used in ECL designs to connect output and mustjoin pins before routing to the remaining pins.

`PIN pinName` Specifies the name of an I/O pin on a net or a subnet.

`PLACED pt orient` Specifies that the pin has a location, but can be moved during automatic layout. You must specify the pin's location and orientation.

`PROPERTY propName propVal` Specifies a numerical or string value for a net property defined in the `PROPERTYDEFINITIONS` statement. The *propName* you specify must match the *propName* listed in the `PROPERTYDEFINITIONS` statement.

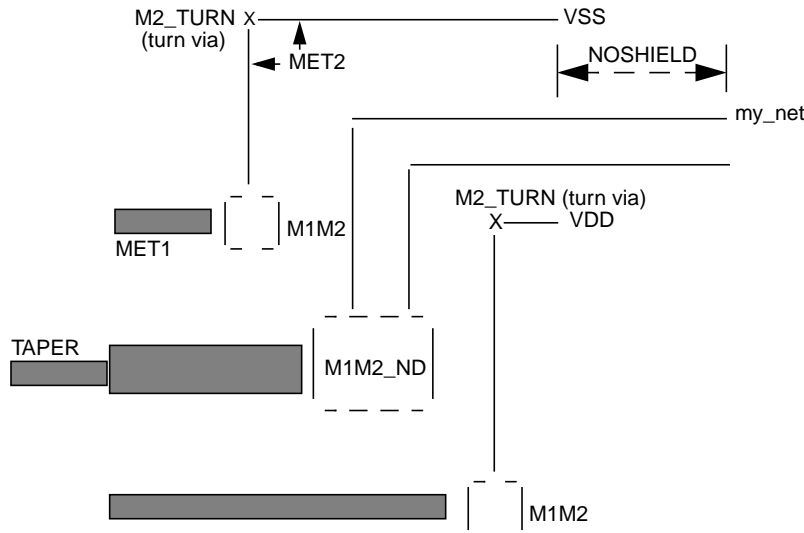
`regularWiring` Specifies the regular physical wiring for the net or subnet. For regular wiring syntax, see ["Regular Wiring Statement"](#) on page 114.

`SHIELDNET shieldNetName` Specifies the name of a special net that shields the regular net

LEF/DEF 5.4 Language Reference

DEF Syntax

being defined. A shield net for a regular net is defined earlier in the DEF file in the `SPECIALNETS` section.



SOURCE {DIST | NETLIST | TEST | TIMING | USER}

Specifies the source of the net. The value of this field is preserved when input to the DEF reader.

Value: Specify one of the following:

DIST	Net is the result of adding cells or creating clock trees.
NETLIST	Net is defined in the DEF file.
TEST	Net is a scanchain.
TIMING	Net represents logical rather than physical change to netlist (as with Clock Tree Generator or placement based synthesis).
USER	Net is user defined.

SUBNET *subnetName*

Names and defines a subnet of the regular net *netName*. A subnet must have at least two pins. The subnet pins can be virtual pins, regular pins, or a combination of virtual and regular pins. A subnet pin cannot be a mustjoin pin.

LEF/DEF 5.4 Language Reference

DEF Syntax

SYNTHESIZED	Specifies that the scan pin has been synthesized. After synthesis, OUTPUT DEF marks the scan pin as SYNTHESIZED and the net as TEST.																
USE {ANALOG CLOCK GROUND POWER RESET SCAN SIGNAL TIEOFF}	<p>Specifies how the net is used.</p> <p>Value: Specify one of the following:</p> <table><tr><td>ANALOG</td><td>Used as a analog signal net.</td></tr><tr><td>CLOCK</td><td>Used as a clock net.</td></tr><tr><td>GROUND</td><td>Used as a ground net.</td></tr><tr><td>POWER</td><td>Used as a power net.</td></tr><tr><td>RESET</td><td>Used as a reset net.</td></tr><tr><td>SCAN</td><td>Used as a scan net.</td></tr><tr><td>SIGNAL</td><td>Used as digital signal net.</td></tr><tr><td>TIEOFF</td><td>Used as a tie-high or tie-low net.</td></tr></table>	ANALOG	Used as a analog signal net.	CLOCK	Used as a clock net.	GROUND	Used as a ground net.	POWER	Used as a power net.	RESET	Used as a reset net.	SCAN	Used as a scan net.	SIGNAL	Used as digital signal net.	TIEOFF	Used as a tie-high or tie-low net.
ANALOG	Used as a analog signal net.																
CLOCK	Used as a clock net.																
GROUND	Used as a ground net.																
POWER	Used as a power net.																
RESET	Used as a reset net.																
SCAN	Used as a scan net.																
SIGNAL	Used as digital signal net.																
TIEOFF	Used as a tie-high or tie-low net.																
VPIN <i>vpinName</i> <i>pt</i> <i>pt</i>	<p>Specifies the name of a virtual pin, and its physical geometry. Virtual pins can be used only in subnets. A SUBNET statement refers to virtual pins by the <i>vpinName</i> specified here. You must define each virtual pin in a + VPIN statement before you can list it in a SUBNET statement.</p>																
WEIGHT <i>weight</i>	<p>Specifies the weight of the net. Automatic layout tools attempt to shorten the lengths of nets with high weights. Do not specify a net weight larger than 10, or assign weights to more than 3 percent of the nets in a design.</p> <p>Note: The net constraints method of controlling net length is preferred over using net weights.</p>																
XTALK <i>class</i>	<p>Specifies the crosstalk class number for the net. If you specify the default value (0), XTALK will not be written to the DEF file.</p> <p><i>Default:</i> 0</p> <p><i>Type:</i> Integer</p> <p><i>Value:</i> 0 to 200</p>																

LEF/DEF 5.4 Language Reference

DEF Syntax

Regular Wiring Statement

```
{+ COVER | + FIXED | + ROUTED | + NOSHIELD}  
  layerName [TAPER | TAPERRULE ruleName] ( x y [value] )  
    {( x * [value] ) | ( * y [value] ) | ( * * [value] ) | viaName} ...  
  [NEW layerName [TAPER | TAPERRULE ruleName] ( x y [value] )  
    {( x * [value] ) | ( * y [value] ) | ( * * [value] ) | viaName} ...  
  ] ...
```

Specifies regular wiring for the net.

COVER	Specifies that the wiring cannot be moved by either automatic layout or interactive commands. If no wiring is specified for a particular net, the net is unrouted. If you specify COVER, you must also specify <i>layerName</i> .
FIXED	Specifies that the wiring cannot be moved by automatic layout, but can be changed by interactive commands. If no wiring is specified for a particular net, the net is unrouted. If you specify FIXED, you must also specify <i>layerName</i> .
<i>layerName</i>	Specifies the layer on which the wire lies. You must specify <i>layerName</i> if you specify COVER, FIXED, ROUTED, or NEW. Specified layers must be routable, and reference to a cut layer generates an error.
NEW <i>layerName</i>	Indicates a new wire segment, that is, that there is no wire segment between the last specified coordinate and the next coordinate. Also specifies the name of the layer on which the new wire lies. Noncontinuous paths can be defined in this manner.
NOSHIELD	Specifies that the last wide segment of the net is not shielded. If the last segment is not shielded, and is tapered, specify TAPER under the LAYER argument, instead of NOSHIELD.
ROUTED	Specifies that the wiring can be moved by the automatic layout tools. If no wiring is specified for a particular net, the net is unrouted. If you specify ROUTED, you must also specify <i>layerName</i> .
TAPER	Specifies that the next contiguous wire segment is created using the default rule.

LEF/DEF 5.4 Language Reference

DEF Syntax

<code>TAPERRULE <i>ruleName</i></code>	Specifies that the next contiguous wire segment is created using the specified nondefault rule.
<code>(<i>x</i> <i>y</i>) (<i>x</i> *), (* <i>y</i>) (* *)</code>	Specifies the coordinate of the form. For more information, see “Specifying Coordinates” on page 115
<code><i>value</i></code>	<p>Specifies how far the wire is extended at the specified point. If the value is omitted, wires extend half the routing width for the layer or rule.</p> <p>You can use the following extension values only on pins:</p> <ul style="list-style-type: none"><input type="checkbox"/> 0<input type="checkbox"/> The WIREEXTENSION value for the layer or rule <p>You can use extension values larger than half the width only before or after a via.</p>
<code><i>viaName</i></code>	Places the specified via at the last specified coordinate. The DEF reader does not detect a mismatch between the current layer and a via.

Specifying Coordinates

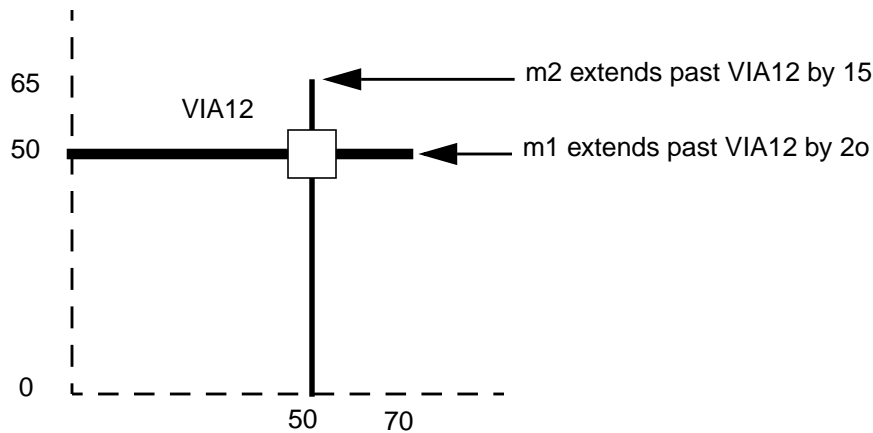
To maximize compactness of the design files, the coordinates allow for the asterisk (*) convention. Here, (*x* *) indicates that the Y coordinate last specified in the wiring

LEF/DEF 5.4 Language Reference

DEF Syntax

specification is used; (* *y*) indicates that the X coordinate last specified is used. Use (* * *value*) to specify a wire extension at a via.

```
ROUTED M1 ( 0 50 ) ( 50 * 20 ) VIA12 ( * * 15 ) ( * 0 )
```



Each coordinate sequence defines a connected orthogonal path through the points. The first coordinate in a sequence must not have an * element.

Because nonorthogonal segments are not allowed, subsequent points in a connected sequence must create orthogonal paths. For example, the following sequence is a valid path:

```
( 100 200 ) ( 200 200 ) ( 200 500 )
```

The following sequence is an equivalent path:

```
( 100 200 ) ( 200 * ) ( * 500 )
```

The following sequence is not valid because it represents a nonorthogonal segment.

```
( 100 200 ) ( 300 500 )
```



Orientation Information

If you specify the pin's placement status, you must specify its location and orientation. A pin can have any of the following orientations:

Value	Definition	Value	Definition
N	North	FN	Flipped north
S	South	FS	Flipped south
W	West	FW	Flipped west

LEF/DEF 5.4 Language Reference

DEF Syntax

Value	Definition	Value	Definition
E	East 	FE	Flipped east 

Examples

The following example defines a virtual pin.

```
+ VPIN M7K.v2 LAYER MET2 ( -10 -10 ) ( 10 10 ) FIXED ( 10 10 )
+ SUBNET M7K.2 ( VPIN M7K.v2 ) ( /PREG_CTRL/I$73/A I )
  NONDEFAULTRULE rule1
  ROUTED MET2 ( 27060 341440 ) ( 26880 * ) ( * 213280 )
  M1M2 ( 95040 * ) ( * 217600 ) ( 95280 * )
  NEW MET1 ( 1920 124960 ) ( 87840 * )
  COVER MET2 ( 27060 341440 ) ( 26880 * )
```

The following example defines a shielded net.

```
NETS 1 ;
- my_net ( I1 CLK ) ( BUF OUT )
+ SHIELDNET VSS
+ SHIELDNET VDD
  ROUTED
  MET2 ( 14000 341440 ) ( 9600 * ) ( * 282400 )
  M1M2 ( 2400 * )
+ NOSHIELD MET2 ( 14100 341440 ) ( 14000 * )
+ TAPER MET1 ( 2400 282400 ) ( 240 * )

END NETS
```

Pins

```
[PINS numPins ;
  [ [- pinName + NET netName]
    [+ SPECIAL]
    [+ DIRECTION {INPUT | OUTPUT | INOUT | FEEDTHRU}]
    [+ USE {SIGNAL | POWER | GROUND | CLOCK | TIEOFF | ANALOG | SCAN | RESET}]
    [+ ANTENNAPINPARTIALMETALAREA value [LAYER layerName] ] ...
    [+ ANTENNAPINPARTIALMETALSIDEAREA value [LAYER layerName] ] ...
    [+ ANTENNAPINPARTIALCUTAREA value [LAYER layerName] ] ...
    [+ ANTENNAPINGATEAREA value [LAYER layerName] ] ...
    [+ ANTENNAPINDIFFAREA value [LAYER layerName] ] ...
    [+ ANTENNAPINMAXAREACAR value LAYER layerName]
    [+ ANTENNAPINMAXSIDEAREACAR value LAYER layerName]
    [+ ANTENNAPINMAXCUTCAR value LAYER layerName]
    [+ LAYER layerName pt pt]
    [+ COVER pt orient | FIXED pt orient | PLACED pt orient]
  ; ] ...
```

LEF/DEF 5.4 Language Reference

DEF Syntax

END PINS]

Defines external pins. Each pin definition assigns a pin name for the external pin and associates the pin name with a corresponding internal net name. The pin name and the net name can be the same.

For Silicon Ensemble, when a block (subchip) is to be created from the design using `OUTPUT LEF BLOCK`, the `PINS` statement includes a physical description of the pin and can include placement information. When the block is placed in the next level of the design, the existing wiring for the given net and any connected pin ports are promoted to a single pin port with the given pin name. If you use the `PINMAP` option of the `OUTPUT LEF BLOCK` command, all pin assignments in the `PINS` table are ignored.

When the design is a chip rather than a block, the `PINS` statement describes logical pins, without placement or physical information. Logical pins allow timing constraints to pass correctly between Silicon Ensemble and other tools.

`ANTENNAPINDIFFAREA value [LAYER layerName]`

Specifies the diffusion (diode) area to which the pin is connected on a layer. If you do not specify *layerName*, the value applies to all layers. This is not necessary for output pins.

Type: Integer

Value: Due to the maximum integer size, you can specify a value up to $2^{31}-1$ in database units. In microns squared, this is $22^{31}-1 / \text{DEFConvertFactor}^2$.

For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

`ANTENNAPINGATEAREA value [LAYER layerName]`

Specifies the gate area to which the pin is connected on a layer. If you do not specify *layerName*, the value applies to all layers. This is not necessary for input pins.

Type: Integer

Value: Due to the maximum integer size, you can specify a value up to $2^{31}-1$ in database units. In microns squared, this is $2^{31}-1 / \text{DEFConvertFactor}^2$.

For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

`ANTENNAPINMAXAREACAR value LAYER layerName`

For hierarchical process antenna effect calculation, specifies the maximum cumulative antenna ratio value, using the metal area

LEF/DEF 5.4 Language Reference

DEF Syntax

below the current pin layer. Use this to calculate the actual cumulative antenna ratio on the pin layer, or the layer above it. For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

Type: Integer

ANTENNAPINMAXCUTCAR *value* LAYER *layerName*

For hierarchical process antenna effect calculation, specifies the maximum cumulative antenna ratio value, using the cut area below the current pin layer. Use this to calculate the actual cumulative antenna ratio for the cuts above the pin layer. For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

Type: Integer

ANTENNAPINMAXSIDEAREACAR *value* LAYER *layerName*

For hierarchical process antenna effect calculation, specifies the maximum cumulative antenna ratio value, using the metal side wall area below the current pin layer. Use this to calculate the actual cumulative antenna ratio on the pin layer, or the layer above it. For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

Type: Integer

ANTENNAPINPARTIALCUTAREA *value* [LAYER *cutLayerName*]

Specifies the partial cut area above the current pin layer and inside the macro cell on a layer. If you do not specify *layerName*, the value applies to all layers. For hierarchical designs, only the cut layer above the I/O pin layer is needed for partial antenna ratio calculation.

Type: Integer

Value: Due to the maximum integer size, you can specify a value up to $2^{31}-1$ in database units. In microns squared, this is $2^{31}-1 / \text{DEFConvertFactor}^2$.

For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

ANTENNAPINPARTIALMETALAREA *value* [LAYER *layerName*]

Specifies the partial metal area connected directly to the I/O pin

LEF/DEF 5.4 Language Reference

DEF Syntax

and the inside of the macro cell on a layer. If you do not specify *layerName*, the value applies to all layers. For hierarchical designs, only the same metal layer as the I/O pin, or the layer above it, is needed for partial antenna ratio calculation.

Type: Integer

Value: Due to the maximum integer size, you can specify a value up to $2^{31}-1$ in database units. In microns squared, this is $2^{31}-1 / \text{DEFConvertFactor}^2$.

For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

ANTENNAPINPARTIALMETALSIDEAREA *value* [LAYER *layerName*]

Specifies the partial metal side wall area connected directly to the I/O pin and the inside of the macro cell on a layer. If you do not specify *layerName*, the value applies to all layers. For hierarchical designs, only the same metal layer as the I/O pin, or the layer above it, is needed for partial antenna ratio calculation.

Type: Integer

Value: Due to the maximum integer size, you can specify a value up to $2^{31}-1$ in database units. In microns squared, this is $2^{31}-1 / \text{DEFConvertFactor}^2$.

For information on process antenna calculation, see [Appendix D, "Calculating and Fixing Process Antenna Violations."](#)

COVER *pt orient*

Specifies the pin's location, orientation, and that it is a part of the cover macro. A COVER pin cannot be moved by automatic tools or by interactive commands. If you specify a placement status for a pin, you must also include a LAYER statement.

DIRECTION { INPUT | OUTPUT | INOUT | FEEDTHRU }

Specifies the pin type.

Value: Specify one of the following:

INPUT Pin that accepts signals coming into the cell.

OUTPUT Pin that drives signals out of the cell.

INOUT Pin that can accept signals going either in or out of the cell.

FEEDTHRU

LEF/DEF 5.4 Language Reference

DEF Syntax

<code>FIXED <i>pt orient</i></code>	Specifies the pin's location, orientation, and that it's location cannot be moved by automatic tools, but can be moved by interactive commands. If you specify a placement status for a pin, you must also include a <code>LAYER</code> statement.																
<code>LAYER <i>layerName pt pt</i></code>	Specifies the routing layer used for the pin, and the pin geometry on that layer. If you specify a placement status for a pin, you must include a <code>LAYER</code> statement.																
<code><i>numPins</i></code>	Specifies the number of pins defined in the <code>PINS</code> section.																
<code><i>pinName</i> + NET <i>netName</i></code>	Specifies the name for the external pin, and the corresponding internal net (defined in <code>NETS</code> or <code>SPECIALNETS</code> statements).																
<code>PLACED <i>pt orient</i></code>	Specifies the pin's location, orientation, and that it's location is fixed, but can be moved during automatic layout. If you specify a placement status for a pin, you must also include a <code>LAYER</code> statement.																
<code>SPECIAL</code>	Identifies the pin as a special pin. Regular routers do not route to special pins. The special router routes special wiring to special pins.																
<code>USE {ANALOG CLOCK GROUND POWER RESET SCAN SIGNAL TIEOFF}</code>	<p>Specifies how the pin is used.</p> <p><i>Value:</i> Specify one of the following:</p> <table><tr><td><code>ANALOG</code></td><td>Pin is used for analog connectivity.</td></tr><tr><td><code>CLOCK</code></td><td>Pin is used for clock net connectivity.</td></tr><tr><td><code>GROUND</code></td><td>Pin is used for connectivity to the chip-level ground distribution network.</td></tr><tr><td><code>POWER</code></td><td>Pin is used for connectivity to the chip-level power distribution network.</td></tr><tr><td><code>RESET</code></td><td>Pin is used as reset pin.</td></tr><tr><td><code>SCAN</code></td><td>Pin is used as scan pin.</td></tr><tr><td><code>SIGNAL</code></td><td>Pin is used for regular net connectivity.</td></tr><tr><td><code>TIEOFF</code></td><td>Pin is used as tie-high or tie-low pin.</td></tr></table>	<code>ANALOG</code>	Pin is used for analog connectivity.	<code>CLOCK</code>	Pin is used for clock net connectivity.	<code>GROUND</code>	Pin is used for connectivity to the chip-level ground distribution network.	<code>POWER</code>	Pin is used for connectivity to the chip-level power distribution network.	<code>RESET</code>	Pin is used as reset pin.	<code>SCAN</code>	Pin is used as scan pin.	<code>SIGNAL</code>	Pin is used for regular net connectivity.	<code>TIEOFF</code>	Pin is used as tie-high or tie-low pin.
<code>ANALOG</code>	Pin is used for analog connectivity.																
<code>CLOCK</code>	Pin is used for clock net connectivity.																
<code>GROUND</code>	Pin is used for connectivity to the chip-level ground distribution network.																
<code>POWER</code>	Pin is used for connectivity to the chip-level power distribution network.																
<code>RESET</code>	Pin is used as reset pin.																
<code>SCAN</code>	Pin is used as scan pin.																
<code>SIGNAL</code>	Pin is used for regular net connectivity.																
<code>TIEOFF</code>	Pin is used as tie-high or tie-low pin.																

LEF/DEF 5.4 Language Reference

DEF Syntax

Extra Physical PIN(S) for One Logical PIN

In the design of place and route blocks, you sometimes want to add extra physical connection points to existing signal ports (usually to enable the signal to be accessed from two sides of the block). One pin has the same name as the net it is connected to. This pin is related to the pins specified in GCF for boundary conditions. Any other pins added to the net must use the following naming conventions.

For extra non-bus bit pin names, use the following syntax:

pinname.extraN *N* is a positive integer, incremented as the physical pins are added

For example:

```
PINS n ;
- a + NET a .... ;
- a.extra1 + NET a ... ;
```

For extra bus bit pin names, use the following syntax:

basename.extraN[index]
basename is simple part of bus bit pin/net name
N is a positive integer, incremented as the physical pins are added.
[index] identifies the specific bit of the bus







For example:

```
PINS n ;
- a[0] + net a[0] ... ;
- a.extra1[0] + net a[0] ... ;
```

Note: The brackets [] are the `BUSBITCHARS` as defined in the DEF header.



Orientation Information

If you specify the pin's placement status, you must specify its location and orientation. A pin can have any of the following orientations:

Value	Definition		Value	Definition	
N	North		FN	Flipped north	
S	South		FS	Flipped south	
W	West		FW	Flipped west	

LEF/DEF 5.4 Language Reference

DEF Syntax

Value	Definition	Value	Definition
E	East 	FE	Flipped east 

Examples

The following example describes a physical I/O pin.

```
# M1 width = 50, track spacing = 120
# M2 width = 60, track spacing = 140

DIEAREA ( -5000 -5000 ) ( 5000 5000 ) ;
TRACKS Y -4900 DO 72 STEP 140 LAYER M2 M1 ;
TRACKS X -4900 DO 84 STEP 120 LAYER M1 M2 ;

PINS 4 ;

# Pin on the left side of the block
- BUSA[0]+ NET BUSA[0] + DIRECTION INPUT
  + LAYER M1 ( -25 0 ) ( 25 165 ) # .5 M1 W + 1 M2 TRACK
  + PLACED ( -5000 2500 ) E ;

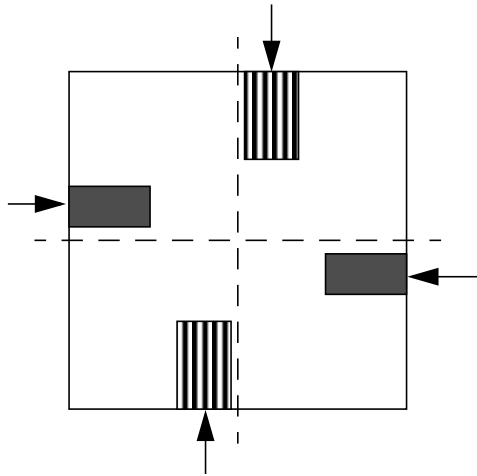
# Pin on the right side of the block
- BUSA[1] + NET BUSA[1] + DIRECTION INPUT
  + LAYER M1 ( -25 0 ) ( 25 165 ) # .5 M1 W + 1 M2 TRACK
  + PLACED ( 5000 -2500 ) W ;

# Pin on the bottom side of the block
- BUSB[0] + NET BUSB[0] + DIRECTION INPUT
  + LAYER M2 ( -30 0 ) ( 30 150 ) # .5 M2 W + 1 M1 TRACK
  + PLACED ( -2100 -5000 ) N ;

# Pin on the top side of the block
- BUSB[1] + NET BUSB[1] + DIRECTION INPUT
  + LAYER M2 ( -30 0 ) ( 30 150 ) # .5 M2 W + 1 M1 TRACK
  + PLACED ( 2100 5000 ) S ;
END PINS
```

LEF/DEF 5.4 Language Reference

DEF Syntax



The following example shows how a logical I/O pin would appear in the DEF file. The pin is defined in Verilog for a chip-level design.

```
module chip (OUT, BUSA, BUSB) ;
    input [0:1] BUSA, BUSB;
    output OUT;
    ....
endmodule
```

The PINS section in DEF looks like this:

```
PINS 5 ;
    - BUSA[0] + NET BUSA[0] + DIRECTION INPUT ;
    - BUSA[1] + NET BUSA[1] + DIRECTION INPUT ;
    - BUSB[0] + NET BUSB[0] + DIRECTION INPUT ;
    - BUSB[1] + NET BUSB[1] + DIRECTION INPUT ;
    - OUT + NET OUT + DIRECTION OUTPUT ;

END PINS
```

Pin Properties

```
[PINPROPERTIES num;
    [- {compName pinName | PIN pinName}
        [+ PROPERTY {propName propVal} ...] ...
    ;] ...

END PINPROPERTIES]
```

Defines pin properties in the design.

LEF/DEF 5.4 Language Reference

DEF Syntax

For Clock Tree Generator, properties are assigned to pins to indicate the clock root and to pass information about triggering edge, excluded pins, and TLF insertion delay to the clock router.

<i>compName pinName</i>	Specifies a component pin. Component pins are identified by the component name and pin name.
<i>num</i>	Specifies the number of pins defined in the PINPROPERTIES section.
PIN <i>pinName</i>	Specifies an I/O pin.
PROPERTY <i>propName propVal</i>	Specifies a numerical or string value for a pin property defined in the PROPERTYDEFINITIONS statement. The <i>propName</i> you specify must match the <i>propName</i> listed in the PROPERTYDEFINITIONS statement.

Example

```
- CORE/g76 CKA + PROPERTY CLOCK "FALLING"
- comp1 A + PROPERTY CLOCK "EXCLUDED"
- rp/regB clk + PROPERTY CLOCK "INSERTION"
```

Property Definitions

```
[PROPERTYDEFINITIONS
  [objectType propName propType [RANGE min max]
   [value | stringValue]
  ;] ...
END PROPERTYDEFINITIONS]
```

Lists all properties used in the design. You must define properties in the PROPERTYDEFINITIONS statement before you can refer to them in other sections of the DEF file.

<i>objectType</i>	Specifies the object type being defined. You can define properties for the following object types: COMPONENT COMPONENTPIN DESIGN
-------------------	---

LEF/DEF 5.4 Language Reference

DEF Syntax

GROUP

NET

REGION

ROW

SPECIALNET

propName Specifies a unique property name for the object type.

propType Specifies the property type for the object type. You can specify one of the following property types:

INTEGER

REAL

STRING

RANGE *min max* Limits real number and integer property values to a specified range.

value | *stringValue* Assigns a numeric value or a name to a DESIGN object.
Note: Assign values to properties for component pins in the PINPROPERTIES section. Assign values to other properties in the section of the LEF file that describes the object to which the property applies.

Examples

The following examples shows a Property Definitions section that defines properties for the design, component pin, and component object types.

```
PROPERTYDEFINITIONS
    DESIGN versionNum INTEGER 12;
    DESIGN title STRING "Buffer";

    COMPONENTPIN count INTEGER RANGE 1 100;

    COMPONENT weight REAL RANGE 1.0 100.0;
    COMPONENT type STRING;

END PROPERTYDEFINITIONS
```

LEF/DEF 5.4 Language Reference

DEF Syntax

When using the ultra placer and ultra router, use the `IGNOREOPTIMIZATION` property keyword to specify nets that are not to be processed by any optimization. The property keyword `IGNOREOPTIMIZATION` needs to be defined in both the `NETS` and `SPECIALNETS` sections, as shown in the following example:

```
PROPERTYDEFINITIONS
...
NET IGNOREOPTIMIZATION STRING ;
SPECIALNET IGNOREOPTIMIZATION STRING ;
...

END PROPERTYDEFINITIONS

SPECIALNETS 200 ;
- sn1 + PROPERTY IGNOREOPTIMIZATION " " ;
- sn2 + PROPERTY IGNOREOPTIMIZATION " " ;
...

END SPECIALNETS

NETS 10000
- n1 + PROPERTY IGNOREOPTIMIZATION " " ;
...

END NETS
```

The ultra placer and ultra router applications require maximum switching frequency per net information to determine wire self heat and hot electron constraints on signals and clocks. The `PROPERTY` keyword is `FREQUENCY`, and needs to be defined in both the `NETS` and `SPECIALNETS` sections, as shown in the following example:

```
PROPERTYDEFINITIONS
...
NET FREQUENCY REAL ;
SPECIALNET FREQUENCY REAL ;
...

END PROPERTYDEFINITIONS

SPECIALNETS 200 ;
- sn1 + PROPERTY FREQUENCY 100.0 ;
- sn2 + PROPERTY FREQUENCY 200.0 ;
...

END SPECIALNETS

NETS 10000
- n1 + PROPERTY FREQUENCY 100.0 ;
...
```

LEF/DEF 5.4 Language Reference

DEF Syntax

```
END NETS
```

The DEF PROPERTYDEFINITIONS section can use the DESIGN object type to point to non-netlist files, such as timing and GCF files, or timing properties derived from GCF constructs. These GCF properties are reserved. Do not create, delete, or modify the properties. Do not delete the referenced files.

```
PROPERTYDEFINITIONS
    DESIGN ACTIVITYFILE STRING "../DATA/VCD/mpeg_big.vcd";
    NET AVGFREQUENCY REAL;
    DESIGN GCFCONSTRAINTS_TIMESTAMP STRING "931884620";
    DESIGN GCFCONSTRAINTS STRING "../DATA/GCF/mpeg.constr.gcf";
    DESIGN GCFCLOCKROOTS STRING "PIN clk";
    DESIGN GCFLIBRARY_TIMESTAMP STRING "932087472";
    DESIGN GCFLIBRARY STRING "../DATA/GCF/mpeg.tlf41.gcf";
END PROPERTYDEFINITIONS
```

Regions

```
[REGIONS numRegions ;
    [- regionName {pt pt} ...
        [+ PROPERTY {propName propVal} ...] ...
    ;] ...
END REGIONS]
```

Defines regions in the design. A region is a physical area to which you can assign a component or group.

numRegions Specifies the number of regions defined in the design.

PROPERTY *propName propVal*
Specifies a numerical or string value for a region property defined in the PROPERTYDEFINITIONS statement. The *propName* you specify must match the *propName* listed in the PROPERTYDEFINITIONS statement.

regionName pt pt Names and defines a region. You define a region as one or more rectangular areas specified by pairs of coordinate points.

For Silicon Ensemble, if a region name is not specified, the system automatically generates a region name based on the value of the environmental variable `Region.Within.Prefix`.

LEF/DEF 5.4 Language Reference

DEF Syntax

The system names the region by appending a number to the `Region.Within.Prefix` string.

You can suppress region names by setting the environmental variable `Output.DEF.Suppress.Region.Name` to `True`.

Example

```
- REGION1 ( 0 0 ) ( 1200 1200 )
+ PROPERTY REGIONORDER 1;
```

Rows

```
[ROW rowName rowType origX origY orient
  { DO numX BY 1 STEP spaceX 0 | DO 1 BY numY STEP 0 spaceY }
  [+ PROPERTY {propName propVal} ...] ... ;] ...
```

Defines rows in the design.

```
DO numX BY 1 STEP spaceX 0 | DO 1 BY numY STEP 0 spaceY
```

Specifies a horizontal or vertical step pattern to create the row.

numX Specifies the number of columns in the pattern.

spaceX Specifies the spacing between the columns.

numY Specifies the number of rows in the pattern.

spaceY Specifies the spacing between the rows.

orient Specifies the orientation of all sites in the row.

origX origY Specifies the location in the design of the first site in the row.

```
PROPERTY propName propVal
```

Specifies a numerical or string value for a row property defined in the `PROPERTYDEFINITIONS` statement. The *propName* you specify must match the *propName* listed in the `PROPERTYDEFINITIONS` statement.

For Silicon Ensemble, `SITE` statements in DEF files from versions earlier than 5.0 are read out as `ROW` statements. To output rows in `SITE` format, set the variable `Output.DEF.45.Format` to `TRUE`.

LEF/DEF 5.4 Language Reference

DEF Syntax

<i>rowName</i>	Specifies the row name used to output the distance constraints between rows.
<i>rowType</i>	Specifies the site type to use for the row. For Silicon Ensemble, this is the same as the name specified by the <code>TYPE</code> option of the <code>ADD ROW</code> command. A standard cell site is a placement location for cells of a particular type. Sites are defined in the LEF file.

Scan Chains

```
[SCANCHAINS numScanChains ;  
  [- chainName  
    [+ COMMONSCANPINS [ ( IN pin ) ] [( OUT pin ) ] ]  
    + START {fixedInComp | PIN} [outPin]  
    [+ FLOATING {floatingComp [ ( IN pin ) ] [ ( OUT pin ) ]} ...]  
    [+ ORDERED  
      {fixedComp [ ( IN pin ) ] [ ( OUT pin ) ]} ...  
    ] ...  
    + STOP {fixedOutComp | PIN} [inPin] ]  
  ;] ...  
END SCANCHAINS]
```

Defines scan chains in the design. Scan chains are a collection of cells that contain both scan-in and scan-out pins. These pins must be defined in the `PIN` statement of LEF file with `USE SCAN`.

<i>chainName</i>	Specifies the name of the scan chain. Each statement in the <code>SCANCHAINS</code> section describes a single scan chain.
------------------	--

`COMMONSCANPINS [(IN pin)] [(OUT pin)]`
Specifies the scan-in and scan-out pins for each component that does not have a scan-in and scan-out pin specified. You must specify either common scan-in and scan-out pins, or individual scan-in and scan-out pins for each component.

`FLOATING {floatingComp [(IN pin)] [(OUT pin)] }`
Specifies the floating list. You can have one or zero floating lists. If you specify a floating list, it must contain at least one component.

For Silicon Ensemble, `TROUTE` can use floating components in any order to synthesize a scan chain. You must specify an in and out pin for each component. If you do not specify the in and out pins here, `TROUTE` uses the pins you specify for the common

LEF/DEF 5.4 Language Reference

DEF Syntax

scan pins. Floating components cannot be shared with other scan chains. TROUTE uses each component only once in synthesizing a scan chain.

ORDERED {*fixedComp* [(IN *pin*)] [(OUT *pin*)] }

Specifies an ordered list. You can specify none or several ordered lists. If you specify an ordered list, you must specify at least two fixed components for each ordered list.

For Silicon Ensemble, TROUTE always synthesizes these components in the same order that you specify them in the list. If you do not specify scan-in and scan-out pins for a component, TROUTE uses the pins you specify for the common scan pins.

Ordered components cannot be shared with other scan chains. TROUTE uses each component only once in synthesizing a scan chain. You do not need to specify the nets for ordered components in the NETS section. TROUTE synthesizes these scan nets in the order you specify.

numScanChains

Specifies the number of scan chains to synthesize.

START {*fixedInComp* | PIN} [*outPin*]

Specifies the start point of the scan chain. You must specify this point. The starting point can be either a component, *fixedInComp*, or an I/O pin, PIN. If you do not specify *outPin*, the router uses the out pin specified for common scan pins.

For Silicon Ensemble, if you do not specify the out pin, TROUTE uses the out pin you specify for the common scan pins. If the scan chain starts at an I/O pin, you must specify the I/O pin name as the out pin.

TROUTE allows different scan chains to share the start and stop scan components. However, TROUTE does not allow sharing of the other scan components by different scan chains.

STOP {*fixedOutComp* | PIN} [*inPin*]

Specifies the end point of the scan chain. You must specify this point. The stop point can be either a component, *fixedOutComp*, or an I/O pin, PIN. If you do not specify *inPin*, the router uses the in pin specified for common scan pins.

LEF/DEF 5.4 Language Reference

DEF Syntax

For Silicon Ensemble, if you do not specify the in pin, TROUTE uses the in pin you specify for the common scan pins. If the scan chain stops at an I/O pin, you must specify the I/O pin name as the in pin. The stop component can be shared with other scan chains.

Scan Chain Rules

Note the following when defining scan chains.

- Each scan-in/scan-out pin pair of adjacent components in the ordered list cannot have different owning nets (unless each net has SOURCE TEST tags or both are tie-off nets).

During execution of INPUT DEF, if the scan net owning the scan-in pin uses a different scan-out pin, a warning message appears suggesting that you use TROUTE DETACH to resolve the net conflict. If you do not run TROUTE DETACH, a problem message appears and the process stops.

- No net (except a tie-off net) can connect a scan-out pin of one component to the scan-in pin of a component in a different scan chain.
- TROUTE is the only tool that can attach or detach synthesized scan pins from or to the scan net. However, you can attach or detach synthesized pins by modifying a library model. You can also attach or detach random logic pins in the synthesized scan nets with incremental INPUT DEF or INPUT DEF ECO.
- For incremental DEF, if you have a COMPONENTS section and a SCANCHAINS section in the same DEF file, the COMPONENTS section must appear before the SCANCHAINS section. If the COMPONENTS section and SCANCHAINS section are in different DEF files, you must read the COMPONENTS section (using INPUT DEF) or load the database before reading the SCANCHAINS section.

Examples

```
Nets 100; #Number of nets resulting after TROUT synthesis.
- SCAN-1 ( C1 SO + SYNTHESIZED )
          ( C4 SI + SYNTHESIZED ) + SOURCE TEST ;
- ...
- N1 ( C3 SO + SYNTHESIZED ) ( C11 SI
    + SYNTHESIZED ) ( AND1 A ) ;
- ...

END NETS
```

LEF/DEF 5.4 Language Reference

DEF Syntax

```
SCANCHAINS 2; #Specified before INPUT DEF
- S1
+ COMMONSCANPINS ( IN SI ) ( OUT SO )
+ START SIPAD OUT
+ FLOATING C1 C2 ( IN D ) ( OUT Q ) C3 C4 C5...CN
+ ORDERED A1 ( OUT Q ) A2 ( IN D ) ( OUT Q ) ...
      AM ( N D )
+ ORDERED B1 B2 ... BL
+ STOP SOPAD IN ;
- S2
```

END SCANCHAINS

```
SCANCHAINS 2 ; #Specified after OUTPUT DEF
- S1
+ START SIPAD OUT
+ FLOATING C1 ( IN SI ) ( OUT SO )
      C2 ( IN D ) ( OUT Q )
      C3 ( IN SI ( OUT SO ) ... CN ( IN SI ) ( OUT SO )
+ ORDERED A1 ( IN SI ) ( OUT Q )
      A2 ( IN D ) ( OUT Q ) ... AM ( IN D ) ( OUT SO )
+ ORDERED B1 ( IN SI ) ( OUT SO )
      B2 ( IN SI ) ( OUT SO ) ...
+ STOP SOPAD IN ;
- S2 ... ;
```

END SCANCHAINS

```
Nets 100; #Number of nets resulting after TROUT synthesis.
- SCAN-1 ( C1 SO + SYNTHESIZED )
      ( C4 SI + SYNTHESIZED ) + SOURCE TEST ;
- ...
- N1 ( C3 SO + SYNTHESIZED ) ( C11 SI
      + SYNTHESIZED ) ( AND1 A ) ;
- ...
```

END NETS

```
SCANCHAINS 2; #Specified before INPUT DEF
- S1
+ COMMONSCANPINS ( IN SI ) ( OUT SO )
+ START SIPAD OUT
+ FLOATING C1 C2 ( IN D ) ( OUT Q ) C3 C4 C5...CN
+ ORDERED A1 ( OUT Q ) A2 ( IN D ) ( OUT Q ) ...
      AM ( N D )
+ ORDERED B1 B2 ... BL
+ STOP SOPAD IN ;
- S2
```

END SCANCHAINS

LEF/DEF 5.4 Language Reference

DEF Syntax

```
SCANCHAINS 2 ; #Specified after OUTPUT DEF
- S1
+ START SIPAD OUT
+ FLOATING C1 ( IN SI ) ( OUT SO )
    C2 ( IN D ) ( OUT Q )
    C3 ( IN SI ( OUT SO ) ... CN ( IN SI ) ( OUT SO )
+ ORDERED A1 ( IN SI ) ( OUT Q )
    A2 ( IN D ) ( OUT Q ) ... AM ( IN D ) ( OUT SO )
+ ORDERED B1 ( IN SI ) ( OUT SO )
    B2 ( IN SI ) ( OUT SO ) ...
+ STOP SOPAD IN ;
- S2 ... ;

END SCANCHAINS
```

Slots

```
[SLOTS numSlots ;
    [- LAYER layerName
        RECT {pt pt} ...
    ;] ...

END SLOTS]
```

Defines the rectangular shapes that form the slotting of the wires in the design. Each slot is defined as an individual rectangle.

<code>LAYER <i>layerName</i></code>	Specifies the layer on which to create slots.
<code><i>numSlots</i></code>	Specifies the number of <code>LAYER</code> statements in the <code>SLOTS</code> statement, <i>not</i> the number of rectangles.
<code>RECT <i>pt pt</i></code>	Specifies the lower left and upper right corner coordinates of the slot geometry.

Examples

The following example defines slots for layers `MET1` and `MET2`.

```
SLOTS 2 ;
- LAYER MET1
    RECT (1000 2000) (1500 4000)
    RECT (2000 2000) (2500 4000)
    RECT (3000 2000) (3500 4000) ;
- LAYER MET2
    RECT (1000 2000) (1500 4000)
```

LEF/DEF 5.4 Language Reference

DEF Syntax

```
RECT (1000 4500) (1500 6500)
RECT (1000 7000) (1500 9000)
RECT (1000 9500) (1500 11500) ;
```

END SLOTS

Special Nets

```
[SPECIALNETS numNets ;
  [- netName [ ( compNameRegExpr pinName [+ SYNTHESIZED] ) ] ...
    [+ WIDTH layerName width] ...
    [+ VOLTAGE volts]
    [specialWiring] ...
    [+ SOURCE {DIST | NETLIST | TIMING | USER}]
    [+ ORIGINAL netName]
    [+ USE {ANALOG | CLOCK | GROUND | POWER | RESET | SCAN | SIGNAL | TIEOFF}]
    [+ PATTERN {BALANCED | STEINER | TRUNK | WIREDLOGIC}]
    [+ ESTCAP wireCapacitance]
    [+ WEIGHT weight]
    [+ PROPERTY {propName propVal} ...] ...
  ;] ...

END SPECIALNETS]
```

Defines netlist connectivity for nets containing special pins. Each specification in the SPECIALNETS statement describes a single net, identified by *netName* and the special pins on the net. These pins are identified by their pin names and corresponding components.

Input parameters for a net can appear in the NETS section or the SPECIALNETS section. In case of conflicting values for an argument, the DEF reader uses the last value encountered for the argument. NETS and SPECIALNETS statements can appear more than once in a DEF file. If a particular net has mixed wiring or pins, specify the special wiring and pins first.

You can also specify the netlist in the COMPONENTS statement. If the netlist is specified in both NETS and COMPONENTS statements, and if the specifications are not consistent, an error message appears. On output, the writer outputs the netlist in either format, depending on the command arguments of the output command.

compNameRegExpr pinName

Specifies the name of a special pin on the net and its corresponding component. You can use a regular expression to specify a set of component names. During evaluation of the regular expression, components that match the expression but do not have a pin named *pinName* are ignored.

ESTCAP wireCapacitance

Specifies the estimated wire capacitance for the net. ESTCAP

LEF/DEF 5.4 Language Reference

DEF Syntax

can be loaded with simulation data to generate net constraints for timing-driven layout.

netName Specifies the name of the net.

ORIGINAL *netName* Specifies the original net partitioned to create multiple nets, including the current net.

PATTERN {BALANCED | STEINER | TRUNK | WIREDLOGIC}
Specifies the routing pattern used for the net.

Default: STEINER

Value: Specify one of the following:

BALANCED Used to minimize skews in timing delays for clock nets.

STEINER Used to minimize net length.

TRUNK Used to minimize delay for global nets.

WIREDLOGIC Used in ECL designs to connect output and mustjoin pins before routing to the remaining pins.

PROPERTY *propName propVal*

Specifies a numerical or string value for a net property defined in the PROPERTYDEFINITIONS statement. The *propName* you specify must match the *propName* listed in the PROPERTYDEFINITIONS statement.

specialWiring Specifies the special wiring for the net. For syntax information, see [Special Wiring Statement](#) on page 137.

SOURCE {DIST | NETLIST | TIMING | USER}

Specifies how the net is created. The value of this field is preserved when input to the DEF reader.

DIST Net is the result of adding cells or creating clock trees.

NETLIST Net is defined in the DEF file.

TIMING Net represents logical rather than physical change to netlist, as with CTGEN or placement-based synthesis.

USER Net is user defined.

LEF/DEF 5.4 Language Reference

DEF Syntax

SYNTHESIZED	Specifies that the scan pin is synthesized. After synthesis, OUTPUT DEF marks the scan pin as SYNTHESIZED and the net as TEST.																
USE {ANALOG CLOCK GROUND POWER RESET SCAN SIGNAL TIEOFF}	<p>Specifies how the net is used.</p> <p>Value: Specify one of the following:</p> <table><tr><td>ANALOG</td><td>Used as a analog signal net.</td></tr><tr><td>CLOCK</td><td>Used as a clock net.</td></tr><tr><td>GROUND</td><td>Used as a ground net.</td></tr><tr><td>POWER</td><td>Used as a power net.</td></tr><tr><td>RESET</td><td>Used as a reset net.</td></tr><tr><td>SCAN</td><td>Used as a scan net.</td></tr><tr><td>SIGNAL</td><td>Used as digital signal net.</td></tr><tr><td>TIEOFF</td><td>Used as a tie-high or tie-low net.</td></tr></table>	ANALOG	Used as a analog signal net.	CLOCK	Used as a clock net.	GROUND	Used as a ground net.	POWER	Used as a power net.	RESET	Used as a reset net.	SCAN	Used as a scan net.	SIGNAL	Used as digital signal net.	TIEOFF	Used as a tie-high or tie-low net.
ANALOG	Used as a analog signal net.																
CLOCK	Used as a clock net.																
GROUND	Used as a ground net.																
POWER	Used as a power net.																
RESET	Used as a reset net.																
SCAN	Used as a scan net.																
SIGNAL	Used as digital signal net.																
TIEOFF	Used as a tie-high or tie-low net.																
VOLTAGE <i>volts</i>	Specifies the voltage for the net as an integer in units of .001 volts. For example, 1.5 v is equal to 1500 in DEF.																
WEIGHT <i>weight</i>	<p>Specifies the weight of the net. Automatic layout tools attempt to shorten the lengths of nets with high weights. Do not specify a net weight larger than 10, or assign weights to more than 3 percent of the nets in a design.</p> <p>Note: The net constraints method of controlling net length is preferred over using net weights.</p>																
WIDTH <i>layerName width</i>	Sets the default width for special wires on the specified layer. If no width is specified for a particular layer, the wire width defaults first to the size of the object picked, and then to the default wire width from the library (the width of regular wires).																

Special Wiring Statement

```
{+ COVER | + FIXED | + ROUTED | + SHIELD shieldNetName}  
  layerName width  
    [+ SHAPE {RING | PADRING | BLOCKRING | STRIPE | FOLLOWPIN  
              | IOWIRE | COREWIRE | BLOCKWIRE | BLOCKAGEWIRE | FILLWIRE}]  
    ( x y ) { ( x * ) | ( * y ) | viaName } ...
```

LEF/DEF 5.4 Language Reference

DEF Syntax

```
[NEW layerName width
  [+ SHAPE {RING | PADRING | BLOCKRING | STRIPE | FOLLOWPIN
            | IOWIRE | COREWIRE | BLOCKWIRE | BLOCKAGEWIRE | FILLWIRE}]
  ( x y ) { ( x * ) | ( * y ) | viaName } ...
] ...
```

Defines the wiring for both routed and shielded nets.

COVER Specifies that the wiring cannot be moved by either automatic layout or interactive commands. If no wiring is specified for a particular net, the net is unrouted. If you specify **COVER**, you must also specify *layerName width*.

FIXED Specifies that the wiring cannot be moved by automatic layout, but can be changed by interactive commands. If no wiring is specified for a particular net, the net is unrouted. If you specify **FIXED**, you must also specify *layerName width*.

layerName width Sets the *width* for wires on the specified layer. You must specify *layerName width* if you specify **COVER**, **FIXED**, **ROUTED**, or **NEW**. Specified layers must be routable, and reference to a cut layer generates an error.

When a via is placed in special wiring, the previously established *width* is used for the wiring in the new layer. To change the *width*, a new path must be specified as follows: **NEW** *layerName width*.

Note: Wire *width* is stored as a half-width in the database. The width value is divided by 2 and truncated. Therefore, wire widths that are specified as odd integers in database precision units are rounded down to the next even integer before being written to the database.

NEW *layerName width*

Indicates a new wire segment, that is, that there is no wire segment between the last specified coordinate and the next coordinate. Also specifies the name of the layer on which the new wire lies, and the width for the wire. Noncontinuous paths can be defined in this manner.

ROUTED Specifies that the wiring can be moved by automatic layout tools. If no wiring is specified for a particular net, the net is unrouted. If

LEF/DEF 5.4 Language Reference

DEF Syntax

you specify `ROUTED`, you must also specify *layerName* *width*.

SHAPE

Specifies a wire with special connection requirements because of its shape. This applies to vias as well as wires.

Value: Specify one of the following:

RING	Used as ring, target for connection
PADRING	Connects padings
BLOCKRING	Connects rings around the blocks
STRIPE	Used as stripe
FOLLOWPIN	Created by FOLLOWPIN
IOWIRE	Connects I/O to target
COREWIRE	Connects end points of FOLLOWPIN to target
BLOCKWIRE	Connects block pin to target
BLOCKAGEWIRE	Created by ADD BLOCKAGE
FILLWIRE	Connects metal fills to power and ground. Represents fill wire that is connected to special net

SHIELD *shieldNetName*

Specifies the name of a regular net to be shielded by the special net being defined.

After describing shielded routing for a net, use `+ROUTED` to return to the routing of the special net being defined.

$(x\ y)\{(x\ *)\ (*\ y)\}$

Specifies the coordinate of the form. For more information, see [“Specifying Coordinates”](#) on page 140

viaName

Specifies the name of the via to place at the last specified coordinate. The DEF reader does not detect a mismatch between the current layer and a via.

LEF/DEF 5.4 Language Reference

DEF Syntax

Specifying Coordinates

To maximize compactness of the design files, the coordinates allow for the asterisk (*) convention. For example, (x *) indicates that the Y coordinate last specified in the wiring specification is used; (* y) indicates that the X coordinate last specified is used.

Each coordinate sequence defines a connected orthogonal path through the points. The first coordinate in a sequence must not have an * element.

Because nonorthogonal segments are not allowed, subsequent points in a connected sequence must create orthogonal paths. For example, the following sequence is a valid path:

```
( 100 200 ) ( 200 200 ) ( 200 500 )
```

The following sequence is an equivalent path:

```
( 100 200 ) ( 200 * ) ( * 500 )
```

The following sequence is not valid because it represents a nonorthogonal segment.

```
( 100 200 ) ( 300 500 )
```

Special Pins and Wiring

Pins that appear in the SPECIALNETS statement are special pins. The regular routers do not route to these pins. The special router routes special wiring to special pins. If you use a component-based format to input the connectivity for the design, special pins to be routed by the special router also must be specified in the SPECIALNETS statement, because pins included in the COMPONENTS statement are considered regular.

For Silicon Ensemble, ADD WIRE creates regular routing for DEF output if the width is default, and no antennae are created. Similarly, ADD WIRE creates special routing for DEF output for nondefault widths and where antennae are created.

The following example inputs connectivity in a component-based format, specifies VDD and VSS pins as special pins, and marks VDD and VSS nets for special routing:

```
COMPONENTS 3 ;
- C1 AND N1 N2 N3 ; - C2 AND N4 N5 N6 ;
- C3 OR N3 N6 N7 ;
END COMPONENTS

SPECIALNETS 2 ;
- VDD ( * VDD ) + WIDTH M1 5 ;
- VSS ( * VSS ) ;
END SPECIALNETS
```

LEF/DEF 5.4 Language Reference

DEF Syntax

Changing Net Representation

For Silicon Ensemble, you can change regular nets appearing in the NETS statement to special with the following command:

```
CHANGE NET netID REPRESENTATION SPECIAL
```

Changing the net does not change pin type. Wires, vias, and pins in the net remain regular. In some cases, verification reports problems such as this one:

```
Pin in net has no used library port
```

because nets that were connected by regular routing have been connected by special routing. Once INPUT DEF reads a DEF file, regular pins cannot be changed to special, and vice versa.

To change regular wiring to special wiring so you can interactively route the net, use CHANGE NET REPRESENTATION and do the following:

- Ensure that all wire widths in the net are special wire widths.
- Ensure that all vias are defined in the Via Rule section of the LEF file (defines vias for special wiring) or the Via Rule Generate section of the LEF file.

If you want to change the net representation from special to regular so that the automatic routers route the net, use CHANGE NET REPRESENTATION and do the following:

- Ensure that all wires in the net are the default or nondefault width for regular wiring. If not, the conversion does not occur.
- Ensure that the vias in the net are defined in the Via section of the LEF file (defines vias for the default regular wiring) and the Nondefault section of the LEF file (defines vias for all regular wiring except the default wiring). If not, the vias are missing after the conversion.

Shielded Routing

If, in a non-routed design, a net has + SHIELDNET attributes, the router adds shielded routing to this net. + NOSHIELD indicates the last wide segment of the net is not shielded. If the last segment is not shielded and is tapered, use the + TAPER keyword instead of + NOSHIELD. For example:

```
+ SHIELDNET VSS      # both sides will be shielded with VSS
+ SHIELDNET VDD      # one side will be shielded with VDD and
+ SHIELDNET VSS      # one side will be shielded with VSS
```

After you add shielded routing to a special net, it has the following syntax:

LEF/DEF 5.4 Language Reference

DEF Syntax

```
+ SHIELD regularNetName
  MET2 regularWidth ( x y )
```

A shield net specified for a regular net should be defined earlier in the DEF file in the SPECIALNETS section. After describing shielded routing for a net, use + ROUTED to return to the routing of the current special net.

For example:

```
SPECIALNETS 2 ;
- VSS
  + ROUTED MET2 200
  ...
+ SHIELD my_net MET2 100 ( 14100 342440 ) ( 13920 * )
  M2_TURN ( * 263200 ) M1M2 ( 2400 * ) ;
- VDD
  + ROUTED MET2 200
  ...
+ SHIELD my_net MET2 100 ( 14100 340440 ) ( 8160 * )
  M2_TURN ( * 301600 ) M1M2 ( 2400 * ) ;

END SPECIALNETS
```

Technology

```
[TECHNOLOGY technologyName ;]
```

Specifies a technology name for the design in the database. In case of a conflict, the previous name remains in effect.

Tracks

```
[TRACKS
  [{X | Y} start DO numtracks STEP space
  [LAYER layerName] ...
;] ...]
```

Defines the routing grid for a standard cell-based design. Typically, the routing grid is generated when the floorplan is initialized. The first track is located at an offset from the placement grid set by the OFFSET value for the layer in the LEF file. The track spacing is the PITCH value for the layer defined in LEF.

DO <i>numTracks</i>	Specifies the number of tracks to create for the grid. You cannot specify 0 <i>numtracks</i> .
---------------------	--

LAYER <i>layerName</i>	Specifies the routing layer used for the tracks. You can specify more than one layer.
------------------------	---

LEF/DEF 5.4 Language Reference

DEF Syntax

<code>STEP <i>space</i></code>	Specifies the spacing between the tracks.
<code>{<i>X</i> <i>Y</i>} <i>start</i></code>	Specifies the location and direction of the first track defined. <i>x</i> indicates vertical lines; <i>y</i> indicates horizontal lines. <i>start</i> is the <i>x</i> or <i>y</i> coordinate of the first line. For example, <code>x 3000</code> creates a set of vertical lines, with the first line going through (3000 0).

Units

`[UNITS DISTANCE MICRONS DEFconvertFactor ;]`

Specifies the convert factor used to convert DEF distance units into LEF distance units.

LEF only supports values of 100, 200, 1000, and 2000 for the LEF convert factor. The following table illustrates the conversion of LEF distance units into database units:

LEFconvertFactor	LEF	Database Units
100	1 micron	100
200	1 micron	200
1000	1 micron	1000
2000	1 micron	2000

For the Ultra Placer, the Ultra Router, and the Preview floorplanner, the DEF convert factor must be the same as the LEF convert factor.

The following table shows the valid pairings of the LEF convert factor and the corresponding DEF convert factor when the Ultra Placer and the Ultra Router are not used:

LEF Convert Factor	DEF Convert Factor
100	100
200	100
200	200
1000	100
1000	1000
2000	100
2000	2000

LEF/DEF 5.4 Language Reference

DEF Syntax

Any other values or combinations of the LEF convert factor and DEF convert factor are not supported in the `UNITS DISTANCE MICRONS` statement.

An incremental DEF should have the same value as a previous DEF. An error message warns you if an incremental DEF has a different value than what is recorded in the database.

Using DEF Units

The following table shows examples of how DEF units are used:

Units	DEF Units	DEF Value Example	Real Value
Time	.001 nanosecond	1500	1.5 nanoseconds
Capacitance	.000001 picofarad	1,500,000	1.5 picofarads
Resistance	.0001 ohm	15,000	1.5 ohms
Power	.0001 milliwatt	15,000	1.5 milliwatts
Current	.0001 milliamp	15,000	1.5 milliamps
Voltage	.001 volt	1500	1.5 volts

The DEF reader assumes divisor factors such that DEF data is given in the database units shown below.

Unit	Database Precision
1 nanosecond	= 1000 DBUs
1 picofarad	= 1,000,000 DBUs
1 ohm	= 10,000 DBUs
1 milliwatt	= 10,000 DBUs
1 milliamperere	= 10,000 DBUs
1 volt	= 1000 DBUs

Version

`VERSION versionNumber ;`

Specifies which version of the DEF syntax is being used.

LEF/DEF 5.4 Language Reference

DEF Syntax

Vias

```
[VIAS numVias ;  
    [- viaName  
        [+ PATTERNNAME patternName ]  
        [+ RECT layerName pt pt] ...  
    ;] ...  
END VIAS]
```

Lists the names and geometry definitions of all generated vias in the design.

PATTERNNAME *patternName*

Specifies the name of the pattern that encodes the cut pattern of the via. You can select either `bitmapped` or `compressed` for the pattern naming convention with the environmental variable `Synthesis.ViaName.Convention`. With the `bitmapped` convention, the upper limit is 64 cuts. With the `compressed` via convention, there is no upper limit for the via cut number. The `compressed` naming convention is recommended because it creates much shorter pattern names. The `bitmapped` naming convention is retained for compatibility with versions 4.3 and earlier. For more information, see [Compressed Via Pattern Encoding Convention](#) on page 166

numVias

Specifies the number of vias listed in the `VIA` statement.

`RECT` *layerName pt pt*

Defines the via geometry for the specified layer . The points are specified with respect to the via origin. In most cases, the via origin is the center of the via bounding box. All geometries for the via, including the cut layers, are output by the DEF writer.

viaName

Specifies the via name. Via names are generated by appending a number after the rule name. Vias are numbered in the order in which they are created.

LEF/DEF 5.4 Language Reference

DEF Syntax

Examples

This appendix contains information about the following topics.

- [LEF](#) on page 147
- [DEF](#) on page 157
- [Scan Chain Synthesis Example](#) on page 162

LEF

```
# DEMO4 CHIP - 1280 ARRAY
NAMECASESENSITIVE ON
    &alias &&area = (73600,74400) (238240,236400) &endalias
    &alias &&core = (85080,85500) (226760,224700) &endalias
    &alias &&m2stripes = srout stripe net vss net vdd layer m2
        width
        320 count 2 pattern 87900 4200 218100
        area &&area core &&core &endalias
    &alias &&m3stripes = srout stripe net vss net vdd layer m3
        width
        600 count 2 pattern 89840 6720 217520
        area &&area core &&core &endalias
    &alias &&powerfollowpins = srout follow net vss net vdd layer
        m1 width 560
        area &&area core &&core &endalias
    &alias &&powerrepair = srout repair net vss net vdd area
        &&area core &&core &endalias
# PLACEMENT SITE SECTION
SITE CORE1 SIZE 67.2 BY 6 ; # GCD of all Y sizes of Macros END
CORE1
SITE IOX SIZE 37.8 BY 444 ; # 151.2 / 4 = 37.8 , 4 sites per pad END IOX
SITE IOY SIZE 436.8 BY 30 ; # 150 / 5 = 30 , 5 sites per pad END IOY
SITE SQUAREBLOCK SIZE 268.8 BY 252 ; END SQUAREBLOCK
SITE I2BLOCK SIZE 672 BY 504 ; END I2BLOCK
SITE LBLOCK SIZE 201.6 BY 168 ; END LBLOCK
SITE CORNER SIZE 436.8 BY 444 ; END CORNER
```

LEF/DEF 5.4 Language Reference

Examples

```
LAYER POLYS TYPE MASTERSLICE ; END POLYS
LAYER PW TYPE MASTERSLICE ; END PW
LAYER NW TYPE MASTERSLICE ; END NW
LAYER PD TYPE MASTERSLICE ; END PD
LAYER ND TYPE MASTERSLICE ; END ND
LAYER CUT01 TYPE CUT ; END CUT01
LAYER M1 TYPE ROUTING ; DIRECTION VERTICAL ; PITCH 5.6 ; WIDTH2.6 ;
    SPACING 1.5 ; END M1
LAYER CUT12 TYPE CUT ; END CUT12
LAYER M2 TYPE ROUTING ; DIRECTION HORIZONTAL ; PITCH 6.0 ;
    WIDTH 3.2 ; SPACING 1.6 ; END M2
LAYER CUT23 TYPE CUT ; END CUT23
LAYER M3 TYPE ROUTING ; DIRECTION VERTICAL ; PITCH 5.6 ; WIDTH 3.6 ;
    SPACING 1.6 ; END M3
LAYER OVERLAP TYPE OVERLAP ; END OVERLAP
VIA C2PW DEFAULT LAYER PW ; RECT -2.0 -2.0 2.0 2.0 ;
    LAYER CUT01 ; RECT -0.6 -0.6 0.6 0.6 ;
    LAYER M1 ; RECT -2.0 -2.0 2.0 2.0 ; END C2PW
VIA C2NW DEFAULT LAYER NW ; RECT -2.0 -2.0 2.0 2.0 ;
    LAYER CUT01 ; RECT -0.6 -0.6 0.6 0.6 ;
    LAYER M1 ; RECT -2.0 -2.0 2.0 2.0 ; END C2NW
VIA C2PD DEFAULT LAYER PD ; RECT -2.0 -2.0 2.0 2.0 ;
    LAYER CUT01 ; RECT -0.6 -0.6 0.6 0.6 ;
    LAYER M1 ; RECT -2.0 -2.0 2.0 2.0 ; END C2PD
VIA C2ND DEFAULT LAYER ND ; RECT -2.0 -2.0 2.0 2.0 ;
    LAYER CUT01 ; RECT -0.6 -0.6 0.6 0.6 ;
    LAYER M1 ; RECT -2.0 -2.0 2.0 2.0 ; END C2ND
VIA C2POLY DEFAULT LAYER POLYS ; RECT -2.0 -2.0 2.0 2.0 ;
    LAYER CUT01 ; RECT -0.6 -0.6 0.6 0.6 ;
    LAYER M1 ; RECT -2.0 -2.0 2.0 2.0 ; END C2POLY
VIA VIA12 DEFAULT LAYER M1 ; RECT -2.0 -2.0 2.0 2.0 ;
    LAYER CUT12 ; RECT -0.7 -0.7 0.7 0.7 ;
    LAYER M2 ; RECT -2.0 -2.0 2.0 2.0 ; END VIA12
VIA VIA23 DEFAULT LAYER M3 ; RECT -2.0 -2.0 2.0 2.0 ;
    LAYER CUT23 ; RECT -0.8 -0.8 0.8 0.8 ;
    LAYER M2 ; RECT -2.0 -2.0 2.0 2.0 ; END VIA23
    SPACING SAMENET CUT01 CUT12 4.0 ;
    SAMENET CUT12 CUT23 4.0 ; END SPACING
VIARULE TURN1 GENERATE LAYER M1 ; DIRECTION VERTICAL ;
    LAYER M1 ; DIRECTION HORIZONTAL ; END TURN1
VIARULE TURN2 GENERATE LAYER M2 ; DIRECTION VERTICAL ;
    LAYER M2 ; DIRECTION HORIZONTAL ; END TURN2
VIARULE TURN3 GENERATE LAYER M3 ; DIRECTION VERTICAL ;
    LAYER M3 ; DIRECTION HORIZONTAL ; END TURN3
VIA VIACENTER12 LAYER M1 ; RECT -4.6 -2.2 4.6 2.2 ;
    LAYER CUT12 ; RECT -3.1 -0.8 -1.9 0.8 ; RECT 1.9 -0.8 3.1 0.8 ;
    LAYER M2 ; RECT -4.4 -2.0 4.4 2.0 ; END VIACENTER12
```

LEF/DEF 5.4 Language Reference

Examples

```
VIA VIATOP12 LAYER M1 ; RECT -2.2 -2.2 2.2 8.2 ;
    LAYER CUT12 ; RECT -0.8 5.2 0.8 6.8 ;
    LAYER M2 ; RECT -2.2 -2.2 2.2 8.2 ; END VIATOP12
VIA VIABOTTOM12 LAYER M1 ; RECT -2.2 -8.2 2.2 2.2 ;
    LAYER CUT12 ; RECT -0.8 -6.8 0.8 -5.2 ;
    LAYER M2 ; RECT -2.2 -8.2 2.2 2.2 ; END VIABOTTOM12
VIA VIALEFT12 LAYER M1 ; RECT -7.8 -2.2 2.2 2.2 ;
    LAYER CUT12 ; RECT -6.4 -0.8 -4.8 0.8 ;
    LAYER M2 ; RECT -7.8 -2.2 2.2 2.2 ; END VIALEFT12
VIA VIARIGHT12 LAYER M1 ; RECT -2.2 -2.2 7.8 2.2 ;
    LAYER CUT12 ; RECT 4.8 -0.8 6.4 0.8 ;
    LAYER M2 ; RECT -2.2 -2.2 7.8 2.2 ; END VIARIGHT12
VIA VIABIGPOWER12 LAYER M1 ; RECT -21.0 -21.0 21.0 21.0 ;
    LAYER CUT12 ; RECT -2.4 -0.8 2.4 0.8 ;
        RECT -19.0 -19.0 -14.2 -17.4 ; RECT -19.0 17.4 -14.2
        19.0 ;
        RECT 14.2 -19.0 19.0 -17.4 ; RECT 14.2 17.4 19.0 19.0 ;
        RECT -19.0 -0.8 -14.2 0.8 ; RECT -2.4 -19.0 2.4 -17.4 ;
        RECT 14.2 -0.8 19.0 0.8 ; RECT -2.4 17.4 2.4 19.0 ;
    LAYER M2 ; RECT -21.0 -21.0 21.0 21.0 ; END VIABIGPOWER12
VIARULE VIALIST12 LAYER M1 ; DIRECTION VERTICAL ; WIDTH 9.0 TO
9.6 ;
    LAYER M2 ; DIRECTION HORIZONTAL ; WIDTH 3.0 TO 3.0 ;
        VIA VIACENTER12 ; VIA VIATOP12 ; VIA VIABOTTOM12 ;
        VIA VIALEFT12 ; VIA VIARIGHT12 ; END VIALIST12
VIARULE VIAGEN12 GENERATE LAYER M1 ; DIRECTION VERTICAL ;
OVERHANG 1.4 ;
    LAYER M2 ; DIRECTION HORIZONTAL ; OVERHANG 1.4 ;
    LAYER CUT12 ; RECT -0.8 -0.8 0.8 0.8 ;
    SPACING 5.6 BY 6.0 ; END VIAGEN12
VIA VIACENTER23 LAYER M3 ; RECT -2.2 -2.2 2.2 2.2 ;
    LAYER CUT23 ; RECT -0.8 -0.8 0.8 0.8 ;
    LAYER M2 ; RECT -2.0 -2.0 2.0 2.0 ; END VIACENTER23
VIA VIATOP23 LAYER M3 ; RECT -2.2 -2.2 2.2 8.2 ;
    LAYER CUT23 ; RECT -0.8 5.2 0.8 6.8 ;
    LAYER M2 ; RECT -2.2 -2.2 2.2 8.2 ; END VIATOP23
VIA VIABOTTOM23 LAYER M3 ; RECT -2.2 -8.2 2.2 2.2 ;
    LAYER CUT23 ; RECT -0.8 -6.8 0.8 -5.2 ;
    LAYER M2 ; RECT -2.2 -8.2 2.2 2.2 ; END VIABOTTOM23
VIA VIALEFT23 LAYER M3 ; RECT -7.8 -2.2 2.2 2.2 ;
    LAYER CUT23 ; RECT -6.4 -0.8 -4.8 0.8 ;
    LAYER M2 ; RECT -7.8 -2.2 2.2 2.2 ; END VIALEFT23
VIA VIARIGHT23 LAYER M3 ; RECT -2.2 -2.2 7.8 2.2 ;
    LAYER CUT23 ; RECT 4.8 -0.8 6.4 0.8 ;
    LAYER M2 ; RECT -2.2 -2.2 7.8 2.2 ; END VIARIGHT23
VIARULE VIALIST23 LAYER M3 ; DIRECTION VERTICAL ; WIDTH 3.6 TO
3.6 ;
```

LEF/DEF 5.4 Language Reference

Examples

```
        LAYER M2 ; DIRECTION HORIZONTAL ; WIDTH 3.0 TO 3.0 ;
          VIA VIACENTER23 ; VIA VIATOP23 ; VIA VIABOTTOM23 ;
          VIA VIALEFT23 ; VIA VIARIGHT23 ; END VIALIST23
VIARULE VIAGEN23 GENERATE LAYER M3 ; DIRECTION VERTICAL ;
  OVERHANG 1.4 ;
  LAYER M2 ; DIRECTION HORIZONTAL ; OVERHANG 1.4 ;
  LAYER CUT23 ; RECT -0.8 -0.8 0.8 0.8 ;
  SPACING 5.6 BY 6.0 ; END VIAGEN23

MACRO CORNER CLASS ENDCAP BOTTOMLEFT ; SIZE 436.8 BY 444 ; SYMMETRY X
Y ; SITE CORNER ;
  PIN VDD SHAPE RING ; DIRECTION INOUT ;
    PORT LAYER M2 ; WIDTH 20 ; PATH 426.8 200 200 200 200 434 ;
    END END VDD
  PIN VSS SHAPE RING ; DIRECTION INOUT ;
    PORT LAYER M2 ; WIDTH 20 ; PATH 100 434 100 100 ; LAYER M1;
    WIDTH 20 ; PATH 100 100 426.8 100 ; END END VSS END
    CORNER

MACRO IN1X class pad ; FOREIGN IN1X ; SIZE 151.2 BY 444 ;
  SYMMETRY X ; SITE IOX ;
  PIN Z DIRECTION OUTPUT ;
    PORT LAYER M1 ; PATH 61.6 444 72.8 444 ; END END Z
  PIN PO DIRECTION OUTPUT ;
    PORT LAYER M1 ; PATH 78.4 444 84.0 444 ; END END PO
  PIN A DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 95.2 444 100.8 444 ; END END A
  PIN PI DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 106.4 444 112 444 ; END END PI
  PIN VDD DIRECTION INOUT ; SHAPE ABUTMENT ;
    PORT LAYER M2 ; WIDTH 20 ; PATH 10 200 141.2 200 ; END
  END VDD
  PIN VSS DIRECTION INOUT ; SHAPE ABUTMENT ;
    PORT LAYER M1 ; WIDTH 20 ; PATH 10 100 141.2 100 ; END
  END VSS END IN1X

MACRO IN1Y EEQ IN1X ; FOREIGN IN1Y ; class pad ;SIZE 436.8 BY 150 ;
  SYMMETRY Y ; SITE IOY ;
  PIN Z DIRECTION OUTPUT ;
    PORT LAYER M2 ; PATH 0 69 0 75 ; END END Z
  PIN PO DIRECTION OUTPUT ;
    PORT LAYER M2 ; PATH 0 81 0 87 ; END END PO
  PIN A DIRECTION INPUT ;
    PORT LAYER M2 ; PATH 0 51 0 57 ; ENDEND A
  PIN PI DIRECTION INPUT ;
    PORT LAYER M2 ; PATH 0 39 0 45 ; END END PI
  PIN VDD DIRECTION INOUT ; SHAPE ABUTMENT ;
```

LEF/DEF 5.4 Language Reference

Examples

```
        PORT LAYER M2 ; WIDTH 20 ; PATH 236.8 10 236.8 140 ; END
    END VDD
    PIN VSS DIRECTION INOUT ; SHAPE ABUTMENT ;
        PORT LAYER M2 ; WIDTH 20 ; PATH 336.8 10 336.8 140 ; END
    END VSS END IN1Y

MACRO FILLER FOREIGN FILLER ; SIZE 67.2 BY 6 ; SYMMETRY X Y R90 ;
    SITE CORE1 ;
    PIN VDD DIRECTION INOUT ; SHAPE ABUTMENT ;
        PORT LAYER M1 ; RECT 45.8 0 55 6 ; END END VDD
    PIN VSS DIRECTION INOUT ; SHAPE ABUTMENT ;
        PORT LAYER M1 ; RECT 12.2 0 21.4 6 ; END END VSS
    OBS LAYER M1 ; RECT 24.1 1.5 43.5 4.5 ; END END FILLER

MACRO INV FOREIGN INVS ; SIZE 67.2 BY 24 ; SYMMETRY X Y ; SITE CORE1 ;
    PIN Z DIRECTION OUTPUT ;
        PORT LAYER M2 ; PATH 30.8 9 42 9 ; END END Z
    PIN A DIRECTION INPUT ;
        PORT LAYER M1 ; PATH 25.2 15 ; END END A
    PIN VDD DIRECTION INOUT ; SHAPE ABUTMENT ;
        PORT LAYER M1 ; WIDTH 5.6 ; PATH 50.4 4.6 50.4 13.4 ; END
    END VDD
    PIN VSS DIRECTION INOUT ; SHAPE ABUTMENT ;
        PORT LAYER M1 ; WIDTH 5.6 ; PATH 16.8 4.6 16.8 13.4 ; END
    END VSS
    OBS LAYER M1 ; RECT 24.1 1.5 43.5 16.5 ; END END INV

MACRO BUF FOREIGN BUFS ; SIZE 67.2 BY 126 ; SYMMETRY X Y ; SITE
CORE1 ;
    PIN Z DIRECTION OUTPUT ;
        PORT LAYER M2 ; PATH 25.2 39 42 39 ; END END Z
    PIN A DIRECTION INPUT ;
        PORT LAYER M1 ; PATH 30.8 33 ; END END A
    PIN VDD DIRECTION INOUT ; SHAPE FEEDTHRU ;
        PORT LAYER M1 ; WIDTH 5.6 ;
        PATH 50.4 4.6 50.4 10.0 56.0 10.0 56.0 115.8 50.4 115.8
        50.4 121.4 ; END END VDD
    PIN VSS DIRECTION INOUT ; SHAPE FEEDTHRU ;
        PORT LAYER M1 ; WIDTH 5.6 ;
        PATH 16.8 4.6 16.8 10.0 11.2 10.0 11.2 115.8 16.8 115.8
        16.8 121.4 ; END END VSS
    OBS LAYER M1 ; RECT 24.1 1.5 43.5 124.5 ; END END BUF

MACRO BIDIR1X FOREIGN BIDIR1X ; class pad ; SIZE 151.2 BY 444 ;
    SYMMETRY X ; SITE IOX ;
    PIN IO DIRECTION INOUT ;
        PORT LAYER M1 ; PATH 61.6 444 67.2 444 ; END END IO
    PIN ZI DIRECTION OUTPUT ;
        PORT LAYER M1 ; PATH 78.4 444 84.0 444 ; END END ZI
```

LEF/DEF 5.4 Language Reference

Examples

```
PIN PO DIRECTION OUTPUT ;
    PORT LAYER M1 ; PATH 95.2 444 100.8 444 ; END END PO
PIN A DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 106.4 444 112.0 444 ; END END A
PIN EN DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 134.4 444 140.0 444 ; END END EN
PIN TN DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 28.0 444 33.6 444 ; END END TN
PIN PI DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 44.8 444 50.4 444 ; END END PI
PIN VDD DIRECTION INOUT ; SHAPE ABUTMENT ;
    PORT LAYER M2 ; WIDTH 20 ; PATH 10 200 141.2 200 ; END END VDD
PIN VSS DIRECTION INOUT ; SHAPE ABUTMENT ;
    PORT LAYER M1 ; WIDTH 20 ; PATH 10 100 141.2 100 ; END
    END VSS END BIDIR1X

MACRO BIDIR1Y EEQ BIDIR1X ; class pad ; FOREIGN BIDIR1Y ; SIZE 436.8
    BY 150 ; SYMMETRY Y ; SITE IOY ;
PIN IO DIRECTION INOUT ;
    PORT LAYER M2 ; PATH 0 69 0 75 ; END END IO
PIN ZI DIRECTION OUTPUT ;
    PORT LAYER M2 ; PATH 0 93 0 99 ; END END ZI
PIN PO DIRECTION OUTPUT ;
    PORT LAYER M2 ; PATH 0 81 0 87 ; END END PO
PIN A DIRECTION INPUT ;
    PORT LAYER M2 ; PATH 0 15 0 21 ; END END A
PIN EN DIRECTION INPUT ;
    PORT LAYER M2 ; PATH 0 27 0 33 ; END END EN
PIN TN DIRECTION INPUT ;
    PORT LAYER M2 ; PATH 0 39 0 45 ; END END TN
PIN PI DIRECTION INPUT ;
    PORT LAYER M2 ; PATH 0 51 0 57 ; END END PI
PIN VDD DIRECTION INOUT ; SHAPE ABUTMENT ;
    PORT LAYER M2 ; WIDTH 20 ; PATH 236.8 10 236.8 140 ; END
    END VDD
PIN VSS DIRECTION INOUT ; SHAPE ABUTMENT ;
    PORT LAYER M2 ; WIDTH 20 ; PATH 336.8 10 336.8 140 ; END
    END VSS END BIDIR1Y

MACRO OR2 FOREIGN OR2S ; SIZE 67.2 BY 42 ; SYMMETRY X Y ; SITE
    CORE1 ;
PIN Z DIRECTION OUTPUT ;
    PORT LAYER M2 ; PATH 25.2 39 42 39 ; END END Z
PIN A DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 25.2 15 ; END END A
PIN B DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 25.2 3 ; END END B
PIN VDD DIRECTION INOUT ; SHAPE FEEDTHRU ;
```


LEF/DEF 5.4 Language Reference

Examples

```
    PORT LAYER M1 ; WIDTH 5.6 ;
    PATH 50.4 4.6 50.4 10.0 ; PATH 50.4 27.4 50.4 37.4 ;
    VIA 50.4 3 C2PW ; VIA 50.4 21 C2PW ; VIA 50.4 33 C2PW ;
    VIA 50.4 39 C2PW ; END END VDD
PIN VSS DIRECTION INOUT ; SHAPE FEEDTHRU ;
    PORT LAYER M1 ; WIDTH 5.6 ; PATH 16.8 4.6 16.8 10.0 ;
    PATH 16.8 27.4 16.8 37.4 ;
    VIA 16.8 3 C2NW ; VIA 16.8 15 C2NW ; VIA 16.8 21 C2NW ;
    VIA 16.8 33 C2NW ; VIA 16.8 39 C2NW ; END END VSS
OBS LAYER M1 ; RECT 24.1 1.5 43.5 40.5 ; END END OR2

MACRO AND2 FOREIGN AND2S ; SIZE 67.2 BY 84 ; SYMMETRY X Y ; SITE
CORE1 ;
PIN Z DIRECTION OUTPUT ;
    PORT LAYER M2 ; PATH 25.2 39 42 39 ; END END Z
PIN A DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 42 15 ; END END A
PIN B DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 42 3 ; END END B
PIN VDD DIRECTION INOUT ; SHAPE ABUTMENT ;
    PORT LAYER M1 ; WIDTH 5.6 ; PATH 50.4 4.6 50.4 79.4 ; END
END VDD
PIN VSS DIRECTION INOUT ; SHAPE ABUTMENT ;
    PORT LAYER M1 ; WIDTH 5.6 ; PATH 16.8 4.6 16.8 79.4 ; END
END VSS
OBS LAYER M1 ; RECT 24.1 1.5 43.5 82.5 ; END END AND2

MACRO DFF3 FOREIGN DFF3S ; SIZE 67.2 BY 210 ; SYMMETRY X Y ; SITE
CORE1 ;
PIN Q DIRECTION OUTPUT ;
    PORT LAYER M2 ; PATH 19.6 99 47.6 99 ; END END Q
PIN QN DIRECTION OUTPUT ;
    PORT LAYER M2 ; PATH 25.2 123 42 123 ; END END QN
PIN D DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 30.8 51 ; END END D
PIN G DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 25.2 3 ; END END G
PIN CD DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 36.4 75 ; END END CD
PIN VDD DIRECTION INOUT ; SHAPE FEEDTHRU ;
    PORT LAYER M1 ; WIDTH 5.6 ; PATH 50.4 4.6 50.4 205.4 ;
END END VDD
PIN VSS DIRECTION INOUT ; SHAPE FEEDTHRU ;
    PORT LAYER M1 ; WIDTH 5.6 ; PATH 16.8 4.6 16.8 205.4 ;
END END VSS
OBS LAYER M1 ; RECT 24.1 1.5 43.5 208.5 ; PATH 8.4 3 8.4 123 ;
    PATH 58.8 3 58.8 123 ; PATH 64.4 3 64.4 123 ; END END DFF3
```

LEF/DEF 5.4 Language Reference

Examples

```
MACRO NOR2 FOREIGN NOR2S ; SIZE 67.2 BY 42 ; SYMMETRY X Y ; SITE
CORE1 ;
PIN Z DIRECTION OUTPUT ;
    PORT LAYER M1 ; PATH 42 33 ; END END Z
PIN A DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 25.2 15 ; END END A
PIN B DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 36.4 9 ; END END B
PIN VDD DIRECTION INOUT ; SHAPE FEEDTHRU ;
    PORT LAYER M1 ; WIDTH 5.6 ; PATH 50.4 4.6 50.4 37.4 ; END
END VDD
PIN VSS DIRECTION INOUT ; SHAPE FEEDTHRU ;
    PORT LAYER M1 ; WIDTH 5.6 ; PATH 16.8 4.6 16.8 37.4 ; END
END VSS
OBS LAYER M1 ; RECT 24.1 1.5 43.5 40.5 ; END END NOR2

MACRO AND2J EEQ AND2 ; FOREIGN AND2SJ ; SIZE 67.2 BY 48 ;
SYMMETRY X Y ; ORIGIN 0 6 ; SITE CORE1 ;
PIN Z DIRECTION OUTPUT ;
    PORT LAYER M2 ; PATH 25.2 33 42 33 ; END END Z
PIN A DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 42 15 ; END END A
PIN B DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 42 3 ; END END B
PIN VDD DIRECTION INOUT ; SHAPE FEEDTHRU ;
    PORT LAYER M1 ; WIDTH 5.6 ; PATH 50.4 -1.4 50.4 37.4 ;
END END VDD
PIN VSS DIRECTION INOUT ; SHAPE FEEDTHRU ;
    PORT LAYER M1 ; WIDTH 5.6 ; PATH 16.8 -1.4 16.8 37.4 ;
END END VSS
OBS LAYER M1 ; RECT 24.1 1.5 43.5 34.5 ; END END AND2J

MACRO SQUAREBLOCK FOREIGN SQUAREBLOCKS ; CLASS RING ; SIZE 268.8
BY 252 ; SITE SQUAREBLOCK ;
PIN Z DIRECTION OUTPUT ;
    PORT LAYER M2 ; PATH 22.8 21 246.0 21 ; END END Z
PIN A DIRECTION OUTPUT ;
    PORT LAYER M2 ; PATH 64.4 33 137.2 33 ;
    PATH 137.2 33 137.2 69 ; PATH 137.2 69 204.4 69 ; END
END A
PIN B DIRECTION INPUT ;
    PORT LAYER M2 ; PATH 22.8 129 246.0 129 ; END END B
PIN C DIRECTION INPUT ;
    PORT LAYER M2 ; PATH 70 165 70 153 ; PATH 70 153 126 153 ;
    END END C
PIN D DIRECTION INPUT ;
    PORT LAYER M2 ; PATH 22.8 75 64.4 75 ; END END D
PIN E DIRECTION INPUT ;
```

LEF/DEF 5.4 Language Reference

Examples

```
    PORT LAYER M2 ; PATH 22.8 87 64.4 87 ; END END E
PIN F DIRECTION INPUT ;
    PORT LAYER M2 ; PATH 22.8 99 64.4 99 ; END END F
PIN G DIRECTION INPUT ;
    PORT LAYER M2 ; PATH 22.8 111 64.4 111 ; END END G
PIN VDD DIRECTION INOUT ; SHAPE RING ;
    PORT LAYER M1 ; WIDTH 3.6 ; PATH 4.0 3.5 4.0 248 ;
    PATH 264.8 100 264.8 248 ; PATH 150 3.5 150 100 ;
    LAYER M2 ; WIDTH 3.6 ; PATH 4.0 3.5 150 3.5 ;
    PATH 150 100 264.8 100 ; PATH 4.0 248 264.8 248 ; END
    END VDD
PIN VSS DIRECTION INOUT ; SHAPE RING ;
    PORT LAYER M1 ; WIDTH 3.6 ; PATH 10 10 10 150 ;
    PATH 100 150 100 200 ; PATH 50 200 50 242 ;
    PATH 258.8 10 258.8 242 ; LAYER M2 ; WIDTH 3.6 ;
    PATH 10 150 100 150 ; PATH 100 200 50 200 ;
    PATH 10 10 258.8 10 ; PATH 50 242 258.8 242 ; END END VSS
OBS LAYER M1 ; RECT 13.8 14.0 255.0 237.2 ; END
    END SQUAREBLOCK

MACRO I2BLOCK FOREIGN I2BLOCKS ; CLASS RING ; SIZE 672 BY 504 ;
    SITE I2BLOCK ;
    PIN Z DIRECTION OUTPUT ;
        PORT LAYER M2 ; PATH 22.8 21 649.2 21 ; END END Z
    PIN A DIRECTION OUTPUT ;
        PORT LAYER M2 ; PATH 22.8 63 154.0 63 ; PATH 154.0 63 154.0
129;
        PATH 154.0 129 447.6 129 ; END END A
    PIN B DIRECTION INPUT ;
        PORT LAYER M2 ; PATH 137.2 423 447.6 423 ; END END B
    PIN C DIRECTION INPUT ;
        PORT LAYER M2 ; PATH 204.4 165 271.6 165 ; END END C
    PIN D DIRECTION INPUT ;
        PORT LAYER M2 ; PATH 204.4 171 271.6 171 ; END END D
    PIN E DIRECTION INPUT ;
        PORT LAYER M1 ; PATH 204.4 213 204.4 213 ; END END E
    PIN F DIRECTION INPUT ;
        PORT LAYER M1 ; PATH 406 249 406 273 ; END END F
    PIN G DIRECTION INPUT ;
        PORT LAYER M1 ; PATH 338.8 249 338.8 273 ; END END G
    PIN H DIRECTION INPUT ;
        PORT LAYER M1 ; PATH 372.4 357 372.4 381 ; END END H
    PIN VDD DIRECTION INOUT ; SHAPE RING ;
        PORT LAYER M1 ; WIDTH 3.6 ; PATH 668 3.5 668 80.5 ;
        PATH 467 80.5 467 465.5 ; PATH 668 465.5 668 500.5 ;
        PATH 4 500.5 4 465.5 ; PATH 138 465.5 138 80.5 ;
        PATH 4 80.5 4 3.5 ; LAYER M2 ; WIDTH 3.6 ; PATH 4 3.5 668 3.5;
```

LEF/DEF 5.4 Language Reference

Examples

```
    PATH 668 80.5 467 80.5 ; PATH 467 465.5 668 465.5 ;
    PATH 668 500.5 4 500.5 ; PATH 4 465.5 138 465.5 ;
    PATH 138 80.5 4 80.5 ; END END VDD
PIN VSS DIRECTION INOUT ; SHAPE RING ;
    PORT LAYER M1 ; WIDTH 3.6 ; PATH 662 10 662 74 ;
    PATH 461 74 461 472 ; PATH 662 472 662 494 ; PATH 10 494 10
        472 ;
    PATH 144 472 144 74 ; PATH 10 74 10 10 ; LAYER M2 ; WIDTH
        3.6 ;
    PATH 10 10 662 10 ; PATH 662 74 461 74 ; PATH 461 472 662
        472 ;
    PATH 662 494 10 494 ; PATH 10 472 144 472 ; PATH 144 74 10
        74 ;
    END END VSS
OBS LAYER M1 ; RECT 14 14 658 70 ; RECT 14 476 658 490 ;
    RECT 148 14 457 490 ; # rectilinear shape description
    LAYER OVERLAP ; RECT 0 0 672 84 ; RECT 134.4 84 470.4 462 ;
    RECT 0 462 672 504 ; END END I2BLOCK
MACRO LBLOCK FOREIGN LBLOCKS ; CLASS RING ; SIZE 201.6 BY 168 ; SITE
    LBLOCK ;
PIN Z DIRECTION OUTPUT ;
    PORT LAYER M2 ; PATH 2.8 15 198.8 15 ; END END Z
PIN A DIRECTION OUTPUT ;
    PORT LAYER M2 ; PATH 2.8 81 137.2 81 ; PATH 137.2 81 137.2
        69 ;
    PATH 137.2 69 198.8 69 ; END END A
PIN B DIRECTION INPUT ;
    PORT LAYER M2 ; PATH 2.8 165 64.4 165 ; END END B
PIN C DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 2.8 93 2.8 105 ; END END C
PIN D DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 64.4 93 64.4 105 ; END END D
PIN E DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 198.8 39 198.8 39 ; END END E
PIN F DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 198.8 45 198.8 45 ; END END F
PIN G DIRECTION INPUT ;
    PORT LAYER M1 ; PATH 2.8 111 2.8 111 ; END END G
    PORT LAYER M2 ; WIDTH 3.6 ; PATH 1.8 27 199.8 27 ; END END VDD
PIN VSS DIRECTION INOUT ;
    PORT LAYER M2 ; WIDTH 3.6 ; PATH 1.8 57 199.8 57 ; END END VSS
OBS LAYER M2 ; RECT 1.0 80 66.2 166.5 ; RECT 1.0 1.5 200.6 23 ;
    RECT 1.0 31 200.6 53 ; RECT 1.0 61 200.6 82.5 ;
    # rectilinear shape description
    LAYER OVERLAP ; RECT 0 0 201.6 84 ; RECT 0 84 67.2 168 ;
    END END LBLOCK
```

LEF/DEF 5.4 Language Reference

Examples

END LIBRARY

DEF

The following example shows a design netlist.

```
DESIGN DEMO4CHIP ;
TECHNOLOGY DEMO4CHIP ;
ARRAY DEMO4 ;
UNITS DISTANCE MICRONS 100 ;
COMPONENTS 243 ;

- CORNER1 CORNER ; - CORNER2 CORNER ; - CORNER3 CORNER ;
- CORNER4 CORNER ; - C01 IN1X ; - C02 IN1Y ; - C04 IN1X ;
- C05 IN1X ; - C06 IN1Y ;
- C07 IN1Y ; - C08 IN1Y ; - C09 IN1Y ; - C10 IN1X ; - C11 IN1X ;
- C13 BIDIR1Y ; - C14 INV ; - C15 BUF ; - C16 BUF ; - C17 BUF ;
- C19 BIDIR1Y ; - C20 INV ; - C21 BUF ; - C22 BUF ; - C23 BUF ;
- C25 BIDIR1Y ; - C26 INV ; - C27 BUF ; - C28 BUF ; - C29 BUF ;
- C31 BIDIR1Y ; - C32 INV ; - C33 BUF ; - C34 BUF ; - C35 BUF ;
- C37 BIDIR1X ; - C39 INV ; - C40 BUF ; - C41 BUF ; - C42 BUF ;
- C44 BIDIR1X ; - C45 INV ; - C46 BUF ; - C47 BUF ; - C48 BUF ;
- C50 BIDIR1Y ; - C51 INV ; - C52 BUF ; - C53 BUF ; - C54 BUF ;
- C56 BIDIR1X ; - C57 INV ; - C58 BUF ; - C59 BUF ; - C60 BUF ;
- D02 BIDIR1X ; - D03 INV ; - D04 BUF ; - D05 BUF ; - D06 BUF ;
- D08 BIDIR1X ; - D09 INV ; - D10 BUF ; - D11 BUF ; - D12 BUF ;
- D14 BIDIR1X ; - D15 INV ; - D16 BUF ; - D17 BUF ; - D19 BUF ;
- D33 BIDIR1Y ; - D34 INV ; - D35 BUF ; - D36 BUF ; - D37 BUF ;
- D39 BIDIR1Y ; - D40 INV ; - D41 BUF ; - D42 BUF ; - D43 BUF ;
- D45 BIDIR1Y ; - D46 INV ; - D47 BUF ; - D48 BUF ; - D49 BUF ;
- D82 OR2 ; - D83 OR2 ; - D84 OR2 ; - D85 OR2 ; - D86 OR2 ;
- D87 OR2 ; - D88 OR2 ; - D89 OR2 ; - D90 OR2 ; - D91 OR2 ;
- D92 OR2 ; - D93 OR2 ;
- E01 AND3 ; - E02 AND3 ; - E03 AND3 ; - E04 AND3 ; - E05 AND3 ;
- E06 AND3 ; - E07 AND3 ; - E08 AND3 ; - E09 AND3 ; - E10 AND3 ;
- E11 AND3 ; - E12 AND3 ; - E13 AND3 ; - E14 AND3 ; - E15 AND3 ;
- E16 AND3 ;
- EE16 IN1X ; - E17 IN1X ; - E18 IN1X ; - E19 IN1X ; - E20 IN1X ;
- E21 IN1X ; - E22 IN1X ; - E23 IN1Y ; - E24 IN1Y ; - E25 IN1Y ;
- E26 INV ; - E27 AND2 ; - E28 AND2 ; - E29 AND2 ; - E30 AND2 ;
- E31 AND2 ; - E32 AND2 ; - E33 OR2 ; - E34 OR2 ; - E35 OR2 ;
- E36 OR2 ; - E37 IN1Y ; - E38A01 DFF3 ; - E38A02 DFF3 ;
- E38A03 DFF3 ; - E38A04 DFF3 ; - E38A05 DFF3 ; - F01 I2BLOCK ;
- F04 OR2 ; - F06 OR2 ; - F07 OR2 ; - F08 OR2 ; - F09 SQUAREBLOCK ;
- F12 LBLOCK ;
- Z14 INV ; - Z15 BUF ; - Z16 BUF ; - Z17 BUF ; - Z20 INV ;
```

LEF/DEF 5.4 Language Reference

Examples

```
- Z21 BUF ; - Z22 BUF ; - Z23 BUF ; - Z26 INV ; - Z27 BUF ;
- Z28 BUF ; - Z29 BUF ; - Z32 INV ; - Z33 BUF ; - Z34 BUF ;
- Z35 BUF ; - Z39 INV ; - Z40 BUF ; - Z41 BUF ; - Z42 BUF ;
- Z45 INV ; - Z46 BUF ; - Z47 BUF ; - Z48 BUF ; - Z51 INV ;
- Z52 BUF ; - Z53 BUF ; - Z54 BUF ; - Z57 INV ; - Z58 BUF ; - Z59 BUF ;
- Z60 BUF ; - Z103 INV ; - Z104 BUF ; - Z105 BUF ; - Z106 BUF ;
- Z109 INV ; - Z110 BUF ; - Z111 BUF ; - Z112 BUF ; - Z115 INV ;
- Z116 BUF ; - Z117 BUF ; - Z119 BUF ; - Z134 INV ; - Z135 BUF ;
- Z136 BUF ; - Z137 BUF ; - Z140 INV ; - Z141 BUF ; - Z142 BUF ;
- Z143 BUF ; - Z146 INV ; - Z147 BUF ; - Z148 BUF ; - Z149 BUF ;
- Z182 OR2 ; - Z183 OR2 ; - Z184 OR2 ; - Z185 OR2 ; - Z186 OR2 ;
- Z187 OR2 ; - Z188 OR2 ; - Z189 OR2 ; - Z190 OR2 ; - Z191 OR2 ;
- Z192 OR2 ; - Z193 OR2 ; - Z201 AND3 ; - Z202 AND3 ; - Z203 AND3 ;
- Z204 AND3 ; - Z205 AND3 ; - Z206 AND3 ; - Z207 AND3 ; - Z208 AND3 ;
- Z209 AND3 ; - Z210 AND3 ; - Z211 AND3 ; - Z212 AND3 ; - Z213 AND3 ;
- Z214 AND3 ; - Z215 AND3 ; - Z216 AND3 ; - Z226 INV ; - Z227 AND2 ;
- Z228 AND2 ; - Z229 AND2 ; - Z230 AND2 ; - Z231 AND2 ; - Z232 AND2 ;
- Z233 OR2 ; - Z234 OR2 ; - Z235 OR2 ; - Z236 OR2 ; - Z38A01 DFF3 ;
- Z38A02 DFF3 ; - Z38A03 DFF3 ; - Z38A04 DFF3 ; - Z38A05 DFF3 ;
```

END COMPONENTS

NETS 222 ;

```
- VDD ( Z216 B ) ( Z215 B ) ( Z214 C ) ( Z214 B )
( Z213 C ) ( Z213 B ) ( Z212 C ) ( Z212 B ) ( Z211 C ) ( Z211 B )
( Z210 C ) ( E23 Z ) ( Z143 Z ) ( Z142 Z ) ( Z141 Z ) ( Z119 Z )
( Z117 Z ) ( Z116 Z ) ( Z106 Z ) ( Z105 Z ) ( Z104 Z ) ( Z34 Z )
( Z33 Z ) ( Z28 Z ) ( Z27 Z ) ( Z22 Z ) ( Z21 Z ) ( Z16 Z )
( Z15 Z ) ( D45 PO ) ( D14 PO ) ( C01 PI ) ( D45 TN ) ( D39 TN )
( D33 TN ) ( D14 TN ) ( D08 TN ) ( D02 TN ) ( C56 TN ) ( C50 TN )
( C44 TN ) ( C37 TN ) ( C31 TN ) ( C25 TN ) ( C19 TN ) ( C13 TN ) ;
- VSS ( Z209 C ) ( Z208 C ) ( Z207 C ) ( Z206 C ) ( Z205 C )
( Z204 C ) ( Z203 C ) ( Z202 C ) ( Z201 C ) ( Z149 Z ) ( Z148 Z )
( Z147 Z ) ( Z137 Z ) ( Z136 Z ) ( Z135 Z ) ( Z112 Z ) ( Z111 Z )
( Z110 Z ) ( Z60 Z ) ( Z59 Z ) ( Z58 Z ) ( Z54 Z ) ( Z53 Z )
( Z52 Z ) ( Z47 Z ) ( Z46 Z ) ( Z41 Z ) ( Z40 Z ) ( E18 Z )
( D49 Z ) ( D43 Z ) ( D45 A ) ( D39 A ) ( D33 A ) ( D14 A )
( D08 A ) ( D02 A ) ( C56 A ) ( C50 A ) ( C44 A ) ( C37 A )
( C31 A ) ( C25 A ) ( C19 A ) ( C13 A ) ; - XX1001 ( Z38A04 G )
( Z38A02 G ) ; - XX100 ( Z38A05 G ) ( Z38A03 G ) ( Z38A01 G ) ;
- XX907 ( Z236 B ) ( Z235 B ) ; - XX906 ( Z234 B ) ( Z233 B ) ;
- XX904 ( Z232 B ) ( Z231 B ) ; - XX903 ( Z230 B ) ( Z229 B ) ;
- XX902 ( Z228 B ) ( Z227 B ) ;
- XX900 ( Z235 A ) ( Z233 A ) ( Z232 A ) ( Z230 A ) ( Z228 A ) ( Z226 A ) ;
- Z38QN4 ( Z38A04 QN ) ( Z210 B ) ; - COZ131 ( Z38A04 Q ) ( Z210 A ) ;
- Z38QN3 ( Z38A03 QN ) ( Z209 B ) ; - COZ121 ( Z38A03 Q ) ( Z209 A ) ;
- Z38QN2 ( Z38A02 QN ) ( Z208 B ) ; - COZ111 ( Z38A02 Q ) ( Z208 A ) ;
- Z38QN1 ( Z38A01 QN ) ( Z207 B ) ; - COZ101 ( Z38A01 Q ) ( Z207 A ) ;
```

LEF/DEF 5.4 Language Reference

Examples

```
- XX901 ( Z236 A ) ( Z234 A ) ( Z231 A ) ( Z229 A ) ( Z227 A ) ( Z226 Z )
  ( Z193 A ) ;
- X415 ( Z149 A ) ( Z148 A ) ( Z147 A ) ( Z146 Z ) ; - X413 ( Z143 A )
  ( Z142 A ) ( Z141 A ) ( Z140 Z ) ;
- X411 ( Z137 A ) ( Z136 A ) ( Z135 A ) ( Z134 Z ) ;
- X405 ( Z119 A ) ( Z117 A ) ( Z116 A ) ( Z115 Z ) ;
- X403 ( Z112 A ) ( Z111 A ) ( Z110 A ) ( Z109 Z ) ;
- X401 ( Z106 A ) ( Z105 A ) ( Z104 A ) ( Z103 Z ) ;
- X315 ( Z60 A ) ( Z59 A ) ( Z58 A ) ( Z57 Z ) ;
- X313 ( Z54 A ) ( Z53 A ) ( Z52 A ) ( Z51 Z ) ;
- DIS051 ( Z216 A ) ( Z48 Z ) ;
- X311 ( Z48 A ) ( Z47 A ) ( Z46 A ) ( Z45 Z ) ;
- DIS041 ( Z215 A ) ( Z42 Z ) ; - X309 ( Z42 A ) ( Z41 A ) ( Z40 A )
  ( Z39 Z ) ;
- X307 ( Z35 A ) ( Z34 A ) ( Z33 A ) ( Z32 Z ) ;
- DIS031 ( Z214 A ) ( Z35 Z ) ; - DIS021 ( Z213 A ) ( Z29 Z ) ;
- X305 ( Z29 A ) ( Z28 A ) ( Z27 A ) ( Z26 Z ) ;
- DIS011 ( Z212 A ) ( Z23 Z ) ;
- X303 ( Z23 A ) ( Z22 A ) ( Z21 A ) ( Z20 Z ) ;
- DIS001 ( Z211 A ) ( Z17 Z ) ;
- X301 ( Z17 A ) ( Z16 A ) ( Z15 A ) ( Z14 Z ) ;
- X1000 ( E38A05 G ) ( E38A03 G ) ( E38A01 G ) ( E37 Z ) ;
- CNTEN ( Z38A05 Q ) ( E38A05 Q ) ( E25 A ) ;
- VIH20 ( E37 PI ) ( E25 PO ) ; - X0907 ( E36 B ) ( E35 B ) ( E25 Z ) ;
- CCLK0 ( F09 A ) ( E24 A ) ; - VIH19 ( E25 PI ) ( E24 PO ) ;
- X0906 ( E34 B ) ( E33 B ) ( E24 Z ) ; - CATH1 ( F09 Z ) ( E23 A ) ;
- VIH18 ( E24 PI ) ( E23 PO ) ; - CRLIN ( F08 Z ) ( E22 A ) ;
- VIH17 ( E23 PI ) ( E22 PO ) ; - X0904 ( E32 B ) ( E31 B ) ( E22 Z ) ;
- NXLIN ( F07 Z ) ( E21 A ) ; - VIH16 ( E22 PI ) ( E21 PO ) ;
- X0903 ( E30 B ) ( E29 B ) ( E21 Z ) ; - RPT1 ( F06 Z ) ( E20 A ) ;
- VIH15 ( E21 PI ) ( E20 PO ) ; - X0902 ( E28 B ) ( E27 B ) ( E20 Z ) ;
- AGISL ( F04 Z ) ( E19 A ) ; - VIH14 ( E20 PI ) ( E19 PO ) ;
- X0900 ( E35 A ) ( E33 A ) ( E32 A ) ( E30 A ) ( E28 A ) ( E26 A )
  ( E19 Z ) ;
- TSTCN ( Z38A05 QN ) ( E38A05 QN ) ( E18 A ) ;
- VIH13 ( E19 PI ) ( E18 PO ) ; - BCLK1 ( F01 A ) ( E17 A ) ;
- VIH12 ( E18 PI ) ( E17 PO ) ; - CLR0 ( F01 Z ) ( EE16 A ) ;
- VIH11 ( E17 PI ) ( EE16 PO ) ; - BCLKX1 ( Z216 C ) ( E17 Z )
  ( E16 C ) ; - CLRX0 ( Z38A05 CD ) ( Z38A03 CD ) ( Z38A01 CD )
  ( Z215 C ) ( E38A05 CD ) ( E38A03 CD ) ( E38A01 CD ) ( EE16 Z )
  ( E15 C ) ; - E38QN4 ( E38A04 QN ) ( E10 B ) ;
- CAX131 ( E38A04 Q ) ( E10 A ) ; - E38QN3 ( E38A03 QN ) ( E09 B ) ;
- CAX121 ( E38A03 Q ) ( E09 A ) ; - E38QN2 ( E38A02 QN ) ( E08 B ) ;
- CAX111 ( E38A02 Q ) ( E08 A ) ; - E38QN1 ( E38A01 QN ) ( E07 B ) ;
- CAX101 ( E38A01 Q ) ( E07 A ) ;
- SDD111 ( Z38A05 D ) ( Z205 Z ) ( E38A05 D ) ( E05 Z ) ;
- SDD121 ( Z38A04 D ) ( Z204 Z ) ( E38A04 D ) ( E04 Z ) ;
```

LEF/DEF 5.4 Language Reference

Examples

```
- X0901 ( E36 A ) ( E34 A ) ( E31 A ) ( E29 A ) ( E27 A ) ( E26 Z )  
  ( D93 A ) ;  
- VIH21 ( Z192 A ) ( E37 PO ) ( D92 A ) ;  
- STRDENB0 ( Z206 B ) ( Z202 B ) ( Z201 B ) ( Z189 B ) ( Z188 B )  
  ( F12 A ) ( E06 B ) ( E02 B ) ( E01 B ) ( D89 B ) ( D88 B ) ;  
- STRDENA0 ( Z202 A ) ( Z201 A ) ( Z183 B ) ( Z182 B ) ( F12 Z )  
  ( F01 H ) ( E02 A ) ( E01 A ) ( D83 B ) ( D82 B ) ;  
- DAB151 ( F12 H ) ( D48 Z ) ; - DAA151 ( F08 B ) ( D47 Z ) ;  
- X0415 ( D49 A ) ( D48 A ) ( D47 A ) ( D46 Z ) ;  
- SDD151 ( Z38A01 D ) ( Z201 Z ) ( E38A01 D ) ( E01 Z ) ( D45 EN ) ;  
- X0414 ( Z146 A ) ( D46 A ) ( D45 ZI ) ; - D151 ( E14 C ) ( D45 IO ) ;  
- DAB141 ( F12 G ) ( D42 Z ) ; - DAA141 ( F08 A ) ( D41 Z ) ;  
- X0413 ( D43 A ) ( D42 A ) ( D41 A ) ( D40 Z ) ;  
- SDD141 ( Z38A02 D ) ( Z202 Z ) ( E38A02 D ) ( E02 Z ) ( D39 EN ) ;  
- VIH60 ( D45 PI ) ( D39 PO ) ; - X0412 ( Z140 A ) ( D40 A ) ( D39 ZI ) ;  
- D141 ( E13 C ) ( D39 IO ) ; - SDI131 ( E16 B ) ( D37 Z ) ;  
- DAB131 ( F12 F ) ( D36 Z ) ; - DAA131 ( F07 B ) ( D35 Z ) ;  
- X0411 ( D37 A ) ( D36 A ) ( D35 A ) ( D34 Z ) ;  
- VIH58 ( Z193 Z ) ( D93 Z ) ( D33 PI ) ;  
- SDD131 ( Z38A03 D ) ( Z203 Z ) ( E38A03 D ) ( E03 Z ) ( D33 EN ) ;  
- VIH59 ( D39 PI ) ( D33 PO ) ; - X0410 ( Z134 A ) ( D34 A ) ( D33 ZI ) ;  
- D131 ( E12 C ) ( D33 IO ) ; - SDI101 ( E15 B ) ( D19 Z ) ; ...  
- X0315 ( C60 A ) ( C59 A ) ( C58 A ) ( C57 Z ) ;  
- SDD071 ( Z211 Z ) ( E11 Z ) ( C56 EN ) ;  
- VIH53 ( Z190 Z ) ( D90 Z ) ( D02 PI ) ( C56 PO ) ;  
- X0314 ( Z57 A ) ( C57 A ) ( C56 ZI ) ;  
- D071 ( E08 C ) ( C56 IO ) ; - SDI061 ( E11 B ) ( C54 Z ) ;  
- DAB061 ( F09 H ) ( C53 Z ) ; - DAA061 ( F04 A ) ( C52 Z ) ;  
- X0313 ( C54 A ) ( C53 A ) ( C52 A ) ( C51 Z ) ;  
- SDD061 ( Z212 Z ) ( E12 Z ) ( C50 EN ) ;  
- VIH52 ( Z189 Z ) ( D89 Z ) ( C56 PI ) ( C50 PO ) ;  
- X0312 ( Z51 A ) ( C51 A ) ( C50 ZI ) ;  
- D061 ( E07 C ) ( C50 IO ) ; - SDI051 ( E16 A ) ( C48 Z ) ;  
- DAB051 ( F09 G ) ( C47 Z ) ; - DAA051 ( F01 G ) ( C46 Z ) ;  
- X0311 ( C48 A ) ( C47 A ) ( C46 A ) ( C45 Z ) ;  
- SDD051 ( Z213 Z ) ( E13 Z ) ( C44 EN ) ;  
- VIH51 ( Z188 Z ) ( D88 Z ) ( C50 PI ) ( C44 PO ) ;  
- X0310 ( Z45 A ) ( C45 A ) ( C44 ZI ) ;  
- D051 ( E06 C ) ( C44 IO ) ; - SDI041 ( E15 A ) ( C42 Z ) ;  
- DAB041 ( F09 F ) ( C41 Z ) ; - DAA041 ( F01 F ) ( C40 Z ) ;  
- X0309 ( C42 A ) ( C41 A ) ( C40 A ) ( C39 Z ) ;  
- SDD041 ( Z214 Z ) ( E14 Z ) ( C37 EN ) ;  
- VIH50 ( Z187 Z ) ( D87 Z ) ( C44 PI ) ( C37 PO ) ;  
- X0308 ( Z39 A ) ( C39 A ) ( C37 ZI ) ;  
- D041 ( E05 C ) ( C37 IO ) ; - SDI031 ( E14 A ) ( C35 Z ) ;  
- DAB031 ( F09 E ) ( C34 Z ) ; - DAA031 ( F01 E ) ( C33 Z ) ;  
- X0307 ( C35 A ) ( C34 A ) ( C33 A ) ( C32 Z ) ;
```


LEF/DEF 5.4 Language Reference

Examples

```
- SDD031 ( Z215 Z ) ( E15 Z ) ( C31 EN ) ;
- VIH49 ( Z186 Z ) ( D86 Z ) ( C37 PI ) ( C31 PO ) ;
- X0306 ( Z32 A ) ( C32 A ) ( C31 ZI ) ;
- D031 ( E04 C ) ( C31 IO ) ; - SDI021 ( E13 A ) ( C29 Z ) ;
- DAB021 ( F09 D ) ( C28 Z ) ; - DAA021 ( F01 D ) ( C27 Z ) ;
- X0305 ( C29 A ) ( C28 A ) ( C27 A ) ( C26 Z ) ;
- SDD021 ( Z216 Z ) ( E16 Z ) ( C25 EN ) ;
- VIH48 ( Z185 Z ) ( D85 Z ) ( C31 PI ) ( C25 PO ) ;
- X0304 ( Z26 A ) ( C26 A ) ( C25 ZI ) ;
- D021 ( E03 C ) ( C25 IO ) ; - SDI011 ( E12 A ) ( C23 Z ) ;
- DAB011 ( F09 C ) ( C22 Z ) ; - DAA011 ( F01 C ) ( C21 Z ) ;
- X0303 ( C23 A ) ( C22 A ) ( C21 A ) ( C20 Z ) ;
- SDD011 ( Z209 Z ) ( E09 Z ) ( C19 EN ) ;
- VIH47 ( Z184 Z ) ( D84 Z ) ( C25 PI ) ( C19 PO ) ;
- X0302 ( Z20 A ) ( C20 A ) ( C19 ZI ) ;
- D011 ( E02 C ) ( C19 IO ) ; - SDI001 ( E11 A ) ( C17 Z ) ;
- DAB001 ( F09 B ) ( C16 Z ) ; - DAA001 ( F01 B ) ( C15 Z ) ;
- X0301 ( Z14 A ) ( C17 A ) ( C16 A ) ( C15 A ) ( C14 Z ) ;
- VIH45 ( Z182 Z ) ( D82 Z ) ( C13 PI ) ;
- SDD001 ( Z210 Z ) ( E10 Z ) ( C13 EN ) ;
- VIH46 ( Z183 Z ) ( D83 Z ) ( C19 PI ) ( C13 PO ) ;
- X0300 ( C14 A ) ( C13 ZI ) ; - D001 ( E01 C ) ( C13 IO ) ;
- CCLKB0 ( Z234 Z ) ( Z189 A ) ( E34 Z ) ( D89 A ) ( C11 A ) ;
- VIH10 ( EE16 PI ) ( C11 PO ) ;
- STRAAA ( Z206 A ) ( E06 A ) ( C11 Z ) ;
- CCLKA0 ( Z233 Z ) ( Z188 A ) ( E33 Z ) ( D88 A ) ( C10 A ) ;
- VIH9 ( C11 PI ) ( C10 PO ) ;
- STRB00 ( Z192 B ) ( D92 B ) ( C10 Z ) ;
- CRLINB1 ( Z232 Z ) ( Z187 A ) ( E32 Z ) ( D87 A ) ( C09 A ) ;
- VIH8 ( C10 PI ) ( C09 PO ) ;
- STRA00 ( Z187 B ) ( D87 B ) ( C09 Z ) ;
- CRLINA1 ( Z231 Z ) ( Z186 A ) ( E31 Z ) ( D86 A ) ( C08 A ) ;
- VIH7 ( C09 PI ) ( C08 PO ) ;
- X10001 ( E38A04 G ) ( E38A02 G ) ( C08 Z ) ;
- NXLINB1 ( Z230 Z ) ( Z185 A ) ( E30 Z ) ( D85 A ) ( C07 A ) ;
- VIH6 ( C08 PI ) ( C07 PO ) ;
- CLRX00 ( Z38A04 CD ) ( Z38A02 CD ) ( E38A04 CD ) ( E38A02 CD )
  ( C07 Z ) ;
- NXLINA1 ( Z229 Z ) ( Z184 A ) ( E29 Z ) ( D84 A ) ( C06 A ) ;
- VIH5 ( C07 PI ) ( C06 PO ) ;
- STRBB0 ( Z205 B ) ( Z193 B ) ( E05 B ) ( D93 B ) ( C06 Z ) ;
- RPTB1 ( Z228 Z ) ( Z183 A ) ( E28 Z ) ( D83 A ) ( C05 A ) ;
- VIH4 ( C06 PI ) ( C05 PO ) ;
- STRAA0 ( Z205 A ) ( Z186 B ) ( E05 A ) ( D86 B ) ( C05 Z ) ;
- RPTA1 ( Z227 Z ) ( Z182 A ) ( E27 Z ) ( D82 A ) ( C04 A ) ;
- VIH3 ( C05 PI ) ( C04 PO ) ;
- STRB0 ( Z204 B ) ( Z203 B ) ( Z191 B ) ( Z190 B ) ( E04 B )
```

LEF/DEF 5.4 Language Reference

Examples

```
( E03 B ) ( D91 B ) ( D90 B ) ( C04 Z ) ;
- CNTENB0 ( Z236 Z ) ( Z191 A ) ( E36 Z ) ( D91 A ) ( C02 A ) ;
- VIH2 ( C04 PI ) ( C02 PO ) ;
- STRA0 ( Z204 A ) ( Z203 A ) ( Z185 B ) ( Z184 B ) ( E04 A )
  ( E03 A ) ( D85 B ) ( D84 B ) ( C02 Z ) ;
- CNTENA0 ( Z235 Z ) ( Z190 A ) ( E35 Z ) ( D90 A ) ( C01 A ) ;
- VIH1 ( C02 PI ) ( C01 PO ) ; - CALCH ( E37 A ) ( C01 Z ) ;
#
```

Scan Chain Synthesis Example

You define the scan chain in the COMPONENTS and SCANCHAINS sections in your DEF file.

```
COMPONENTS 100 ;
- SIN MUX ;
- SOUT PAD ;
- C1 SDFF ;
- C2 SDFF ;
- C3 SDFF ;
- C4 SDFF ;
- B1 BUF ;
- A1 AND ; ...
END COMPONENTS

NETS 150 ;
- N1 (C1 SO) (C3 SI) ;
- N2 (C3 SO) (A1 A) ; ...
END NETS
```

You do not need to define any scan nets in the NETS section. This portion of the NETS section shows the effect of the scan chain process on existing nets that use components you specify in the SCANCHAINS section.

```
SCANCHAINS 1 ;
- SC
  + COMMONSCANPINS (IN SI) (OUT SO)
  + START SIN Z2
  + FLOATING C1 C2 C3
  + ORDERED C4 B1 (IN A) (OUT Q) ;
  + STOP SOUT A ;
END SCANCHAINS
```

Because components C1, C2, and C3 are floating, TROUTE SCANCHAIN can synthesize them in any order in the chain. TROUTE synthesizes ordered components (C4 and B1) in the order you specify.

LEF/DEF 5.4 Language Reference

Examples

LEF/DEF 5.4 Language Reference

Examples

Via Pattern Encoding Convention

This appendix contains information about the following topics.

- [Overview](#) on page 165
- [Bitmapped Via Pattern Encoding Convention](#) on page 165
- [Compressed Via Pattern Encoding Convention](#) on page 166

Overview

This appendix explains

- The bitmapped pattern encoding convention for generated vias
- The compressed via pattern encoding convention
- How the compressed pattern encoding convention maps to the bitmapped pattern encoding convention

You can choose the naming convention you want with the following environmental variable:

```
Synthesis.ViaName.Convention  
string [bitmapped, compressed]  
Default: compressed  
expert  
Specifies the encoding convention to use for generated vias.
```

The compressed option creates much shorter via pattern entries in the DEF file. Use the bitmapped option only if you need to maintain compatibility to versions 4.3 and earlier.

Bitmapped Via Pattern Encoding Convention

The bitmapped via pattern encoding represents the cut pattern as hexadecimal digits.

LEF/DEF 5.4 Language Reference

Via Pattern Encoding Convention

Name Form

A symmetric via has equal metal on both sides of the center in both directions. You can use `Synthesis.Delimiter` to set the separator (default value is “ - ”). Generated symmetric vias have the following form:

```
viaRuleName-viaSizeX-viaSizeY-cutPattern
```

An asymmetric via has unequal metal in either direction. You can use `Synthesis.Asymmetric.Delimiter` to set the separator (default value is “ | ”). Generated asymmetric vias have the following form:

```
viaRuleName|minX|minY|maxX|maxY|offsetX|offsetY|cutPattern
```

The *offsetX* and *offsetY* values express the offset between the via center and the origin.

Cut Representation

The following example shows the cut pattern, the binary bits, and the hexadecimal representation of a bitmapped via.

000000	111111 = 3F
0 000	100111 = 27
000000	111111 = 3F
00	110000 = 30

Compressed Via Pattern Encoding Convention

The compressed via pattern encoding convention uses both hexadecimal (base 16) and double-decimal (base 20) digits to represent the cut patterns. The compressed via pattern encoding convention generates much shorter pattern name entries because it uses the double-decimal digits to reduce repetitive patterns and characters.

Bitmapped:

```
viaRuleName-viaSizeX-viaSizeY-cutPattern
```

Compressed:

```
viaRuleName-viaSizeX-viaSizeY-Rp_cutPattern_Rh
```

Rp Number of times the following cut pattern is repeated

LEF/DEF 5.4 Language Reference

Via Pattern Encoding Convention

Rh Number of times the preceding hexadecimal character is repeated

Rp and *Rh* are expressed as double-decimal (base 20) digits.

Character Sets

The following shows the hexadecimal and double-decimal digit character sets. The hexadecimal character set uses alpha characters A to F. The double-decimal character set uses the remaining alpha characters, G to Z.

Hexadecimal Digit Character Set

Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Character	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Double-Decimal Digit Character Set (Base 20)

Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Character	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Rules for Double-Decimal Digit Character Set

The double-decimal digit character set follows these rules:

- The cut pattern repetition count, *Rp*, is in front of the repeated cut pattern. The compressed convention uses *Rp* encoding if the cut pattern repeats two or more times.
- The hexadecimal character repetition count, *Rh*, follows the repeated hexadecimal character. The compressed convention uses *Rh* encoding if the preceding hexadecimal character repeats three or more times.

Example 1

The following shows how a bitmapped via pattern name maps to the same name in compressed form.

- Bitmapped via pattern (340 characters):

VIAGEN12-250-252.4-1FFFFFFFFFFFFFFF- . . . -1FFFFFFFFFFFFFFF

← Repeats 23 times →

LEF/DEF 5.4 Language Reference

Via Pattern Encoding Convention

- Compressed via pattern (24 characters):

VIAGEN12_250_252.4_IJ1FS

Number of times the following pattern repeats itself. IJ means that the following cut pattern, 1F, repeats 23 times.



The number of times the preceding character repeats. S means that the preceding character, F, repeats 12 times.

Example 2

- Bitmapped via pattern name (86 characters):

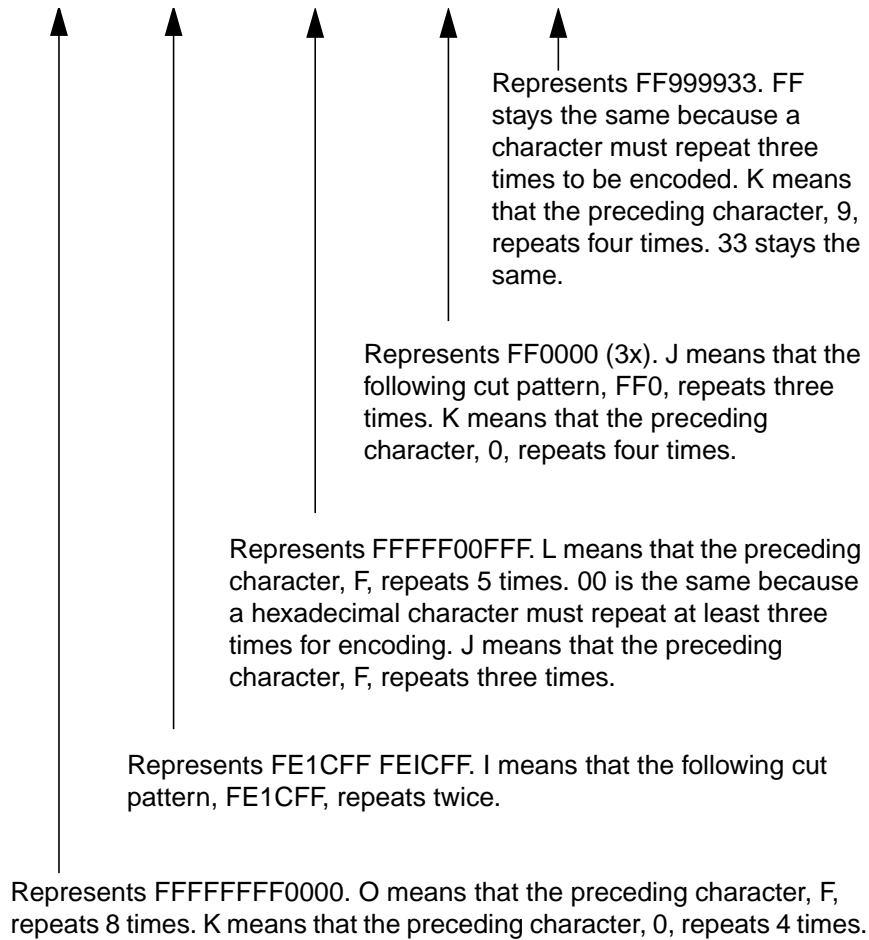
VIAGEN12-250-252.4-FFFFFFFFF0000-FE1CFF-FE1CFF-FFFFFF00FFF-
FF0000-FF0000-FF0000-FF999933

- Compressed via name (51 characters):

LEF/DEF 5.4 Language Reference

Via Pattern Encoding Convention

VIAGEN12_250_252.4_F00K_IFE1CFF_FL00FJ_JFF0K_FF9K33



LEF/DEF 5.4 Language Reference

Via Pattern Encoding Convention

Optimizing LEF Technology for Place and Route

This appendix contains the following information.

- [Overview](#)
- [Guidelines for No Wire Extension at Pin](#) on page 172
- [Guidelines for Routing Pitch](#) on page 172
- [Guidelines for Wide Metal Spacing](#) on page 175
- [Guidelines for Wire Extension at Vias](#) on page 176
- [Guidelines for Default Vias](#) on page 178
- [Guidelines for Stack Vias \(MAR Vias\) and Samenet Spacing](#) on page 180
- [Example of an Optimized LEF Technology File](#) on page 184

Overview

This appendix provides guidelines for defining the optimized technology section in the LEF file to get the best performance using Cadence[®] place-and-route tools, especially Cadence Ultra Router. The LEF syntax shown is based on Silicon Ensemble[®] Place-and-Route 5.2 or newer.

For the following guidelines, the preferred routing direction for `metal1` and all other odd metal layers is horizontal. The preferred routing direction for `metal2` and all other even metal layers is vertical. Standard cells are arranged in horizontal rows.

This appendix discusses the following LEF statements.

```
NOWIREEXTENSIONATPIN { ON | OFF }
```

```
LAYER layerName  
    TYPE ROUTING ;  
    PITCH distance ;
```

LEF/DEF 5.4 Language Reference

Optimizing LEF Technology for Place and Route

```
    WIDTH defWidth ;
    SPACING minSpacing [RANGE minwidth maxwidth] ;
    WIREEXTENSION value ;

END layerName

VIA viaName DEFAULT
    [TOPSTACKONLY]
    LAYER layerName RECT pt pt ; ...

END viaName

SPACING
    SAMENET
        layerName layerName minSpace [STACK] ;

END SPACING
```

Guidelines for No Wire Extension at Pin

By default, wires extend one half of the default routing width at pins. To prevent this wire extension at pins when the pin width is narrower than the wire width, use the following statement in the LEF file.

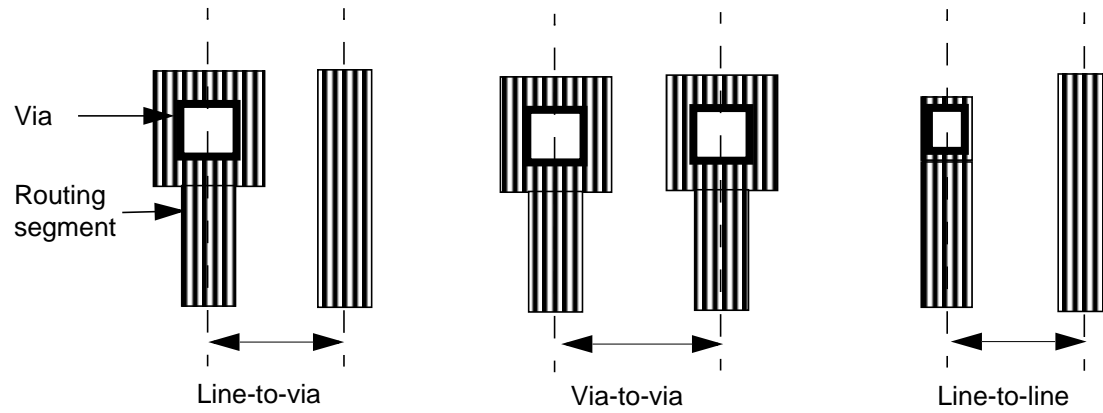
```
NOWIREEXTENSIONATPIN ON ;
```

To control the wire extension at vias, double-check the via size and metal enclosure rule. Then use the `WIREEXTENSION` statement in the appropriate LAYER section.

Guidelines for Routing Pitch

The following is a summary for choosing the right pitch for an existing design library. For detailed information on determining routing pitch, refer to the [Cadence Abstract Generator User Guide](#).

Pitch Measurement

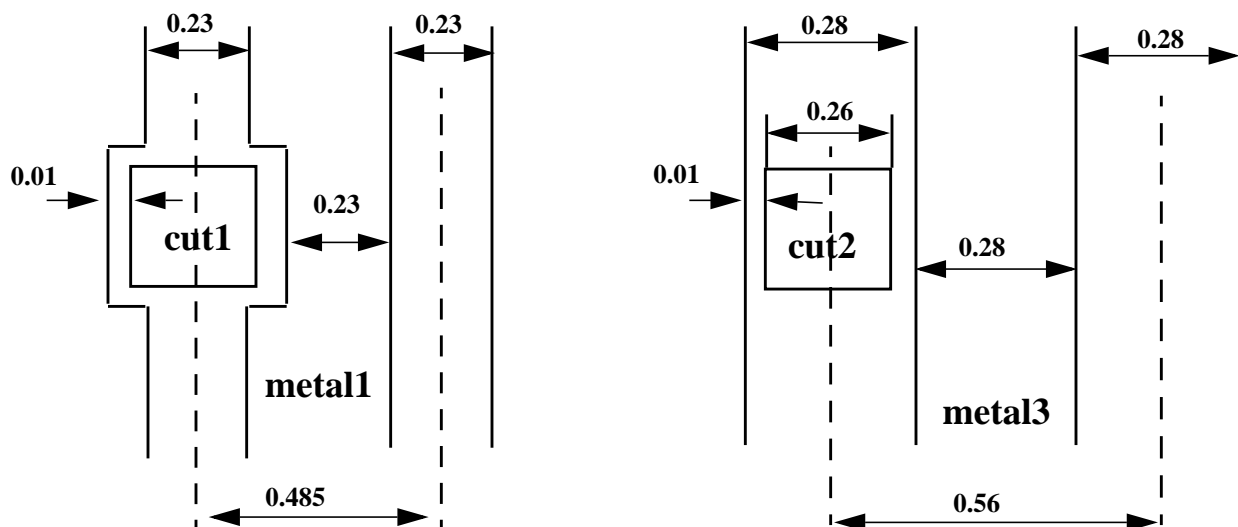


LEF/DEF 5.4 Language Reference

Optimizing LEF Technology for Place and Route

DESIGN RULE No. 1

W.1	Minimum width of <code>metal1</code> = 0.23 μ m
S.1	Minimum space between two <code>metal1</code> regions = 0.23 μ m
W.2	Minimum and maximum width of <code>cut1</code> = 0.26 μ m
E.1	Minimum extension of <code>metal1</code> beyond <code>cut1</code> = 0.01 μ m
W.3	Minimum width of <code>metal3</code> = 0.28 μ m
S.2	Minimum space between two <code>metal3</code> regions = 0.28 μ m
W.4	Minimum and maximum width of <code>cut2</code> = 0.26 μ m
E.2	Minimum extension of <code>metal1</code> beyond <code>cut2</code> = 0.01 μ m



Although the minimum `metal1` routing pitch is 0.485 μ m from the design rule, you should use 0.56 μ m instead, to match the `metal3` routing pitch in the same preferred direction.

LEF Construct No. 1

```
LAYER metal1
  TYPE ROUTING ;
  WIDTH 0.23 ;
```

LEF/DEF 5.4 Language Reference

Optimizing LEF Technology for Place and Route

```
SPACING 0.23 ;
PITCH 0.56 ;
DIRECTION HORIZONTAL ;

END metall
```

```
LAYER metal3
    TYPE ROUTING ;
    WIDTH 0.28 ;
    SPACING 0.28 ;
    PITCH 0.56 ;
    DIRECTION HORIZONTAL ;

END metal3
```

Recommendations

- Use line-to-via spacing for both the horizontal and vertical direction.
- Allow diagonal vias with the routing pitch.
- Align the routing pitch for `metall` and `metal2`, with the pins inside the standard cells.
- Have uniform routing pitch in the same preferred direction. The pitch ratio should be 2 - 3 or 1 - 2. It is better to define the `metall` pitch larger than necessary in order to achieve a 1 - 1 ratio because the `metall` width is usually smaller the `metal2` and `metal3` widths.

Pitch Recommendations for Library Development

- All pins should be on the grid, and only those portions of the pins that are accessible to the router should be modeled as pins. For example, 45 degree pin geometry.
- The height of the cell should be the even multiple of the `metall` pitch, and the width of the cell should be the even multiple of the `metal2` pitch.
- The blockage modeling, especially for `metall`, should be simplified as much as possible. For example, it is very common for the entire area within the cell boundary to be obstructed in `metall`, so use a single rectangular blockage instead of many small blockages.

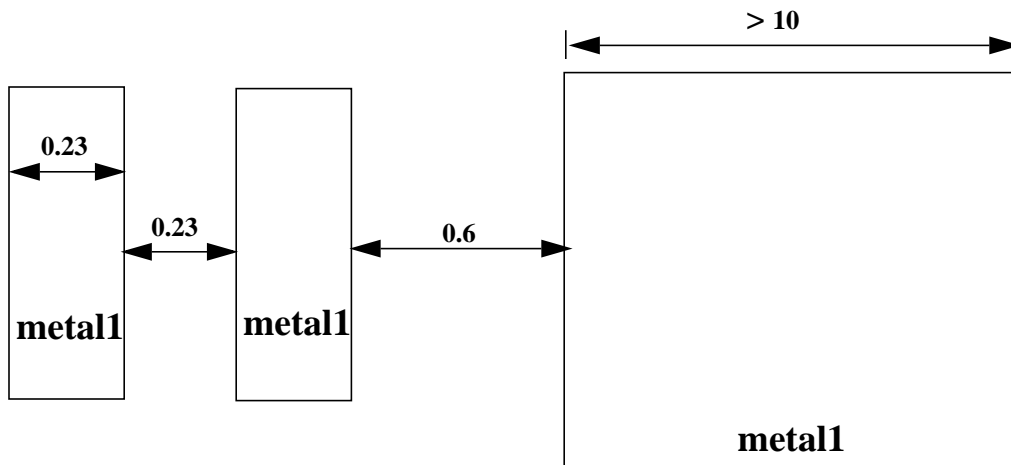
Guidelines for Wide Metal Spacing

The `SPACING` statement in the LEF `LAYER` section is applied to both regular and special wires. You can use the Cadence® ultra router option `frouteUseRangeRule` to determine

which objects to check against the `SPACING RANGE` statement. The default checks both pin and obstruction.

DESIGN RULE No. 2

- S.1 Minimum space between two `metal1` regions = 0.23 μm
- S.2 Minimum space between metal lines with one or both metal line width and length are greater than 10 μm = 0.6 μm



LEF CONSTRUCT No. 2

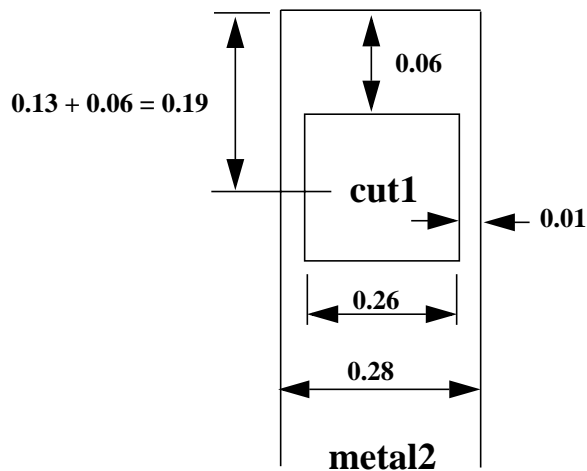
```
LAYER metal1
    WIDTH 0.23 ;
    SPACING 0.23 ;
    SPACING 0.6 RANGE 10.002 1000 ;
END metal1
```

Guidelines for Wire Extension at Vias

The following guidelines are for wire extension at vias.

DESIGN RULE No. 3

- W.1 Minimum and maximum width of `cut1` = 0.26 um
W.2 Minimum width of `metal2` = 0.28 um
E.1 Minimum extension of `metal2` beyond `cut1` = 0.01 um
E.2 Minimum extension of `metal2` end-of-line region beyond `cut1` = 0.06 um



LEF CONSTRUCT No. 3

```
LAYER metal2
  TYPE ROUTING ;
  WIDTH 0.28 ;
  SPACING 0.28 ;
  PITCH 0.56 ;
  WIREEXTENSION 0.19 ;
  DIRECTION VERTICAL ;

END metal2

VIA via23 DEFAULT
  LAYER metal2 ;
  RECT -0.14 -0.14 0.14 0.14 ;          # Use square via
  LAYER cut2 ;
```

LEF/DEF 5.4 Language Reference

Optimizing LEF Technology for Place and Route

```
RECT -0.13 -0.13 0.13 0.13 ;  
LAYER metal3 ;  
RECT -0.14 -0.14 0.14 0.14 ;           # Use square via  
END via23
```

Recommendations

- Use the `WIREEXTENSION` statement instead of defining multiple vias because the width of the `metal2` in `cut1` is the same as the default routing width of the `metal2` layer.
- Define the `DEFAULT VIA` as a square via.

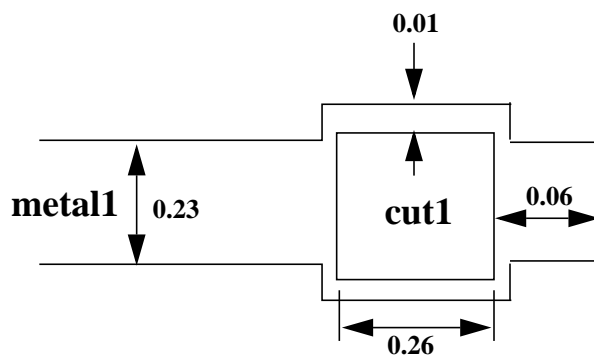
Guidelines for Default Vias

The following guidelines are for default vias.

DESIGN RULE No. 4

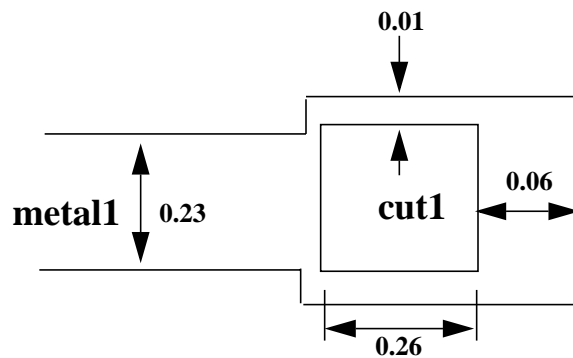
- W.1 Minimum width of `metal1` = 0.23 μ m
- W.2 Minimum and maximum width of `cut1` = 0.26 μ m
- E.1 Minimum extension of `metal1` beyond `cut1` = 0.01 μ m
- E.2 Minimum extension of `metal1` end-of-line region beyond `cut1` = 0.06 μ m

Case A:



Use WIREEXTENSION and
square DEFAULT VIA

Case B:



Use Horizontal and Vertical DEFAULT VIAS

LEF CONSTRUCT No. 4 (Case B)

```

LAYER metal1
    TYPE ROUTING ;
    WIDTH 0.23 ;
    SPACING 0.23 ;
    PITCH 0.56 ;
    DIRECTION HORIZONTAL ;

END metal1

VIA via12_H DEFAULT
    LAYER metal1 ;
    RECT -0.19 -0.14 0.19 0.14 ;           # metal1 end-of-line
    
```

LEF/DEF 5.4 Language Reference

Optimizing LEF Technology for Place and Route

```
        extension 0.6 in both directions
    LAYER cut1 ;
        RECT -0.13 -0.13 0.13 0.13 ;
    LAYER metal2 ;
        RECT -0.14 -0.14 0.14 0.14 ;

END vial2_H

VIA vial2_V DEFAULT
    LAYER metall ;
        RECT -0.14 -0.19 0.14 0.19 ;           # metall end-of-line
        extension 0.6 in both directions
    LAYER cut1 ;
        RECT -0.13 -0.13 0.13 0.13 ;
    LAYER metal2 ;
        RECT -0.14 -0.14 0.14 0.14 ;

END vial2_V
```

Recommendations

- If the width of the end-of-line metal extension is the same as the default metal routing width, as in Case A, use the `WIREEXTENSION` statement in the LEF `LAYER` section, and define a square via in the `DEFAULT VIA` section.
- If the width of the end-of-line metal extension is the same as the width of the via metal, as in Case B, define one horizontal `DEFAULT VIA` and one vertical `DEFAULT VIA` to cover the required metal extension area in both preferred and non-preferred routing directions. Do not use the `WIREEXTENSION` statement in the LEF `LAYER` section.

Guidelines for Stack Vias (MAR Vias) and Samenet Spacing

The following guidelines are for stack vias (minimum area rule) and `SAMENET SPACING`.

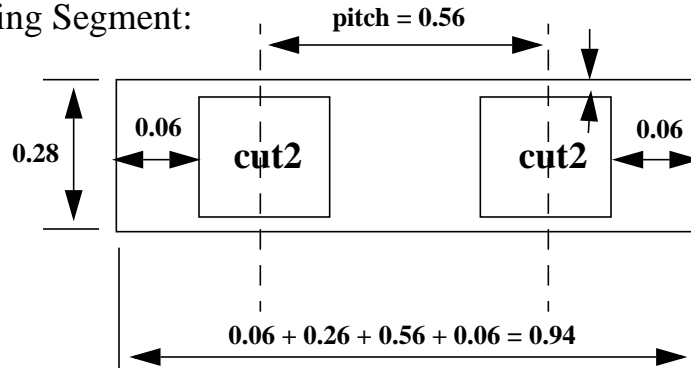
DESIGN RULE No. 5

- | | |
|-----|---|
| W.1 | Minimum width of <code>metal2</code> = 0.28 um |
| W.2 | Minimum and maximum width of <code>cut2</code> = 0.26 um |
| E.1 | Minimum extension of <code>metal2</code> beyond <code>cut2</code> = 0.01 um |
| A.1 | Minimum area of <code>metal2</code> = 0.2025 um |
| C.1 | <code>Cut2</code> can be fully or partially stacked on <code>cut1</code> , contact or any combination |
| W.1 | Minimum width of <code>metal3</code> = 0.28 um |
| W.2 | Minimum and maximum width of <code>cut3</code> = 0.26 um |
| E.1 | Minimum extension of <code>metal2</code> beyond <code>cut3</code> = 0.01 um |
| A.1 | Minimum area of <code>metal3</code> = 0.2025 um |
| C.1 | <code>Cut3</code> can be fully or partially stacked on <code>cut2</code> , <code>cut1</code> , contact or any combination |

LEF/DEF 5.4 Language Reference

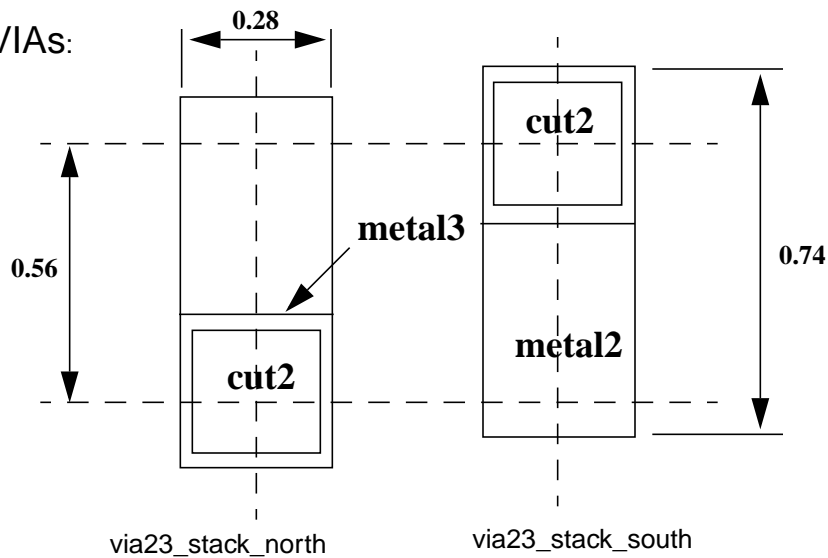
Optimizing LEF Technology for Place and Route

Default Routing Segment:



Minimum routing area of metal3 = $0.28 \times 0.94 = 0.2632 > 0.2.25$ (MAR)

MAR VIAs:



LEF CONSTRUCT No. 5

```
VIA via23_stack_north DEFAULT TOPOFSTACKONLY
  LAYER metal2 ;
    RECT -0.14 -0.14 0.14 0.6 ;    # MAR = 0.28 x 0.74
  LAYER cut2 ;
```

LEF/DEF 5.4 Language Reference

Optimizing LEF Technology for Place and Route

```
        RECT -0.13 -0.13 0.13 0.13 ;
    LAYER metal3 ;
        RECT -0.14 -0.14 0.14 0.14 ;

END via23_stack_north


VIA via23_stack_south DEFAULT TOPOFSTACKONLY
    LAYER metal2 ;
        RECT -0.14 -0.6 0.14 0.14 ;    # MAR = 0.28 x 0.74
    LAYER cut2 ;
        RECT -0.13 -0.13 0.13 0.13 ;
    LAYER metal3 ;
        RECT -0.14 -0.14 0.14 0.14 ;

END via23_stack_south


VIA via34_stack_east DEFAULT TOPOFSTACKONLY
    LAYER metal3 ;
        RECT -0.14 -0.14 0.6 0.14 ;    # MAR = 0.28 x 0.74
    LAYER cut3 ;
        RECT -0.13 -0.13 0.13 0.13 ;
    LAYER metal4 ;
        RECT -0.14 -0.14 0.14 0.14 ;

END via34_stack_east


VIA via34_stack_west DEFAULT TOPOFSTACKONLY
    LAYER metal3 ;
        RECT -0.6 -0.14 0.14 0.14 ;    # MAR = 0.28 x 0.74
    LAYER cut3 ;
        RECT -0.13 -0.13 0.13 0.13 ;
    LAYER metal4 ;
        RECT -0.14 -0.14 0.14 0.14 ;

END via34_stack_west


SPACING
    SAMENET metal2 metal2 0.28  STACK ;
    SAMENET metal3 metal3 0.28  STACK ;

    SAMENET  cut2  cut2  0.26 ;
    SAMENET  cut3  cut3  0.26 ;

    SAMENET  cut1  cut2  0.0 STACK ;
    SAMENET  cut2  cut3  0.0 STACK ;

END SPACING
```

Recommendations

- The minimum metal routing segment (two vias between one pitch grid) with or without end-of-line metal extension should automatically satisfy the minimum area rule.
- If vias are stackable, create the `TOPSTACKONLY` vias with a rectangular shape blocking only one neighboring grid for both sides of the preferred routing direction. In other words, one north oriented and one south oriented for vertical-preferred routing layers, and one east oriented and one west oriented for horizontal-preferred routing layers.
- Use slightly larger dimensions for the via size to make them an even number, so they snap to the manufacturing grids.
- The `STACK` keyword in the `SAMENETSPACING` statements only allows vias to be fully overlapped (stacked) by `SROUTE` commands. To allow vias to be partially overlapped, set the environment variable `SROUTE.ALLOWOVERLAPINSTACKVIA` to `TRUE`.
- The `metal1` layer does not require a `MAR` via because all `metal1` pins should satisfy the minimum area rules.

Example of an Optimized LEF Technology File

```
VERSION 5.2 ;

NAMECASESENSITIVE ON ;

BUSBITCHARS "[]" ;

UNITS
    DATABASE MICRONS 100 ;
END UNITS

LAYER metal1
    TYPE ROUTING ;
    WIDTH 0.23 ;
    SPACING 0.23 ;
    SPACING 0.6 RANGE 10.02 1000 ;
    PITCH 0.56 ;
    DIRECTION HORIZONTAL ;
END metal1

LAYER cut1
    TYPE CUT ;
END cut1
```


LEF/DEF 5.4 Language Reference

Optimizing LEF Technology for Place and Route

```
LAYER metal2
    TYPE ROUTING ;
    WIDTH 0.28 ;
    SPACING 0.28 ;
    SPACING 0.6 RANGE 10.02 1000 ;
    PITCH 0.56 ;
    WIREEXTENSION 0.19 ;
    DIRECTION VERTICAL ;
```

```
END metal2
```

```
LAYER cut2
    TYPE CUT ;
```

```
END cut2
```

```
LAYER metal3
    TYPE ROUTING ;
    WIDTH 0.28 ;
    SPACING 0.28 ;
    SPACING 0.6 RANGE 10.02 1000 ;
    PITCH 0.56 ;
    WIREEXTENSION 0.19 ;
    DIRECTION HORIZONTAL ;
```

```
END metal3
```

```
LAYER cut3
    TYPE CUT ;
```

```
END cut3
```

```
LAYER metal4
    TYPE ROUTING ;
    WIDTH 0.28 ;
    SPACING 0.28 ;
    SPACING 0.6 RANGE 10.02 1000 ;
    PITCH 0.56 ;
    WIREEXTENSION 0.19 ;
    DIRECTION VERTICAL ;
```

```
END metal4
```

LEF/DEF 5.4 Language Reference

Optimizing LEF Technology for Place and Route

```
LAYER cut4
    TYPE CUT ;
END cut4
```

```
LAYER metal5
    TYPE ROUTING ;
    WIDTH 0.28 ;
    SPACING 0.28 ;
    SPACING 0.6 RANGE 10.02 1000 ;
    PITCH 0.56 ;
    WIREEXTENSION 0.19 ;
    DIRECTION HORIZONTAL ;
END metal5
```

```
LAYER cut5
    TYPE CUT ;
END cut5
```

```
LAYER metal6
    TYPE ROUTING ;
    WIDTH 0.44 ;
    SPACING 0.46 ;
    SPACING 0.6 RANGE 10.02 1000 ;
    PITCH 1.12 ;
    DIRECTION VERTICAL ;
END metal6
```

start DEFAULT VIA

```
VIA vial2_H DEFAULT
    LAYER metall ;
        RECT -0.19 -0.14 0.19 0.14 ;      # metall end-of-line ext 0.6
    LAYER cut1 ;
        RECT -0.13 -0.13 0.13 0.13 ;
    LAYER metal2 ;
        RECT -0.14 -0.14 0.14 0.14 ;
END vial2_H
```

```
VIA vial2_V DEFAULT
    LAYER metall ;
        RECT -0.14 -0.19 0.14 0.19 ;      # metall end-of-line ext 0.6
    LAYER cut1 ;
```

LEF/DEF 5.4 Language Reference

Optimizing LEF Technology for Place and Route

```
        RECT -0.13 -0.13 0.13 0.13 ;
    LAYER metal2 ;
        RECT -0.14 -0.14 0.14 0.14 ;
END via12_V
```

```
VIA via23 DEFAULT
    LAYER metal2 ;
        RECT -0.14 -0.14 0.14 0.14 ;
    LAYER cut2 ;
        RECT -0.13 -0.13 0.13 0.13 ;
    LAYER metal3 ;
        RECT -0.14 -0.14 0.14 0.14 ;
END via23
```

```
VIA via34 DEFAULT
    LAYER metal3 ;
        RECT -0.14 -0.14 0.14 0.14 ;
    LAYER cut3 ;
        RECT -0.13 -0.13 0.13 0.13 ;
    LAYER metal4 ;
        RECT -0.14 -0.14 0.14 0.14 ;
END via34
```

```
VIA via45 DEFAULT
    LAYER metal4 ;
        RECT -0.14 -0.14 0.14 0.14 ;
    LAYER cut4 ;
        RECT -0.13 -0.13 0.13 0.13 ;
    LAYER metal5 ;
        RECT -0.14 -0.14 0.14 0.14 ;
END via45
```

```
VIA via56_H DEFAULT
    LAYER metal5 ;
        RECT -0.24 -0.19 0.24 0.19 ;
    LAYER cut5 ;
        RECT -0.18 -0.18 0.18 0.18 ;
    LAYER metal6 ;
        RECT -0.27 -0.27 0.27 0.27 ;
END via56_H
```

LEF/DEF 5.4 Language Reference

Optimizing LEF Technology for Place and Route

```
VIA via56_V DEFAULT
  LAYER metal5 ;
    RECT -0.19 -0.24 0.19 0.24 ;
  LAYER cut5 ;
    RECT -0.18 -0.18 0.18 0.18 ;
  LAYER metal6 ;
    RECT -0.27 -0.27 0.27 0.27 ;

END via56_V
```

end DEFAULT VIA

start STACK VIA

```
VIA via23_stack_north DEFAULT TOPOFSTACKONLY
  LAYER metal2 ;
    RECT -0.14 -0.14 0.14 0.6 ;    # MAR = 0.28 x 0.74
  LAYER cut2 ;
    RECT -0.13 -0.13 0.13 0.13 ;
  LAYER metal3 ;
    RECT -0.14 -0.14 0.14 0.14 ;

END via23_stack_north
```

```
VIA via23_stack_south DEFAULT TOPOFSTACKONLY
  LAYER metal2 ;
    RECT -0.14 -0.6 0.14 0.14 ;    # MAR = 0.28 x 0.74
  LAYER cut2 ;
    RECT -0.13 -0.13 0.13 0.13 ;
  LAYER metal3 ;
    RECT -0.14 -0.14 0.14 0.14 ;

END via23_stack_south
```

```
VIA via34_stack_east DEFAULT TOPOFSTACKONLY
  LAYER metal3 ;
    RECT -0.14 -0.14 0.6 0.14 ;    # MAR = 0.28 x 0.74
  LAYER cut3 ;
    RECT -0.13 -0.13 0.13 0.13 ;
  LAYER metal4 ;
    RECT -0.14 -0.14 0.14 0.14 ;

END via34_stack_east
```

```
VIA via34_stack_west DEFAULT TOPOFSTACKONLY
  LAYER metal3 ;
    RECT -0.6 -0.14 0.14 0.14 ;    # MAR = 0.28 x 0.74
```

LEF/DEF 5.4 Language Reference

Optimizing LEF Technology for Place and Route

```
LAYER cut3 ;
    RECT -0.13 -0.13 0.13 0.13 ;
LAYER metal4 ;
    RECT -0.14 -0.14 0.14 0.14 ;
END via34_stack_west

VIA via45_stack_north DEFAULT TOPOFSTACKONLY
    LAYER metal4 ;
        RECT -0.14 -0.14 0.14 0.6 ;    # MAR = 0.28 x 0.74
    LAYER cut4 ;
        RECT -0.13 -0.13 0.13 0.13 ;
    LAYER metal5 ;
        RECT -0.14 -0.14 0.14 0.14 ;
END via45_stack_north

VIA via45_stack_south DEFAULT TOPOFSTACKONLY
    LAYER metal4 ;
        RECT -0.14 -0.6 0.14 0.14 ;    # MAR = 0.28 x 0.74
    LAYER cut4 ;
        RECT -0.13 -0.13 0.13 0.13 ;
    LAYER metal5 ;
        RECT -0.14 -0.14 0.14 0.14 ;
END via45_stack_south

VIA via56_stack_east DEFAULT TOPOFSTACKONLY
    LAYER metal5 ;
        RECT -0.19 -0.19 0.35 0.19 ;    # MAR = 0.38 x 0.54
    LAYER cut5 ;
        RECT -0.18 -0.18 0.18 0.18 ;
    LAYER metal6 ;
        RECT -0.27 -0.27 0.27 0.27 ;
END via56_stack_east

VIA via56_stack_west DEFAULT TOPOFSTACKONLY
    LAYER metal5 ;
        RECT -0.35 -0.19 0.19 0.19 ;    # MAR = 0.38 x 0.54
    LAYER cut5 ;
        RECT -0.18 -0.18 0.18 0.18 ;
    LAYER metal6 ;
        RECT -0.27 -0.27 0.27 0.27 ;
END via56_stack_west

### end STACK VIA ###
```

LEF/DEF 5.4 Language Reference

Optimizing LEF Technology for Place and Route

SPACING

```
SAMENET metal1 metal1 0.23  STACK ;
SAMENET metal2 metal2 0.28  STACK ;
SAMENET metal3 metal3 0.28  STACK ;
SAMENET metal4 metal4 0.28  STACK ;
SAMENET metal5 metal5 0.28  STACK ;
SAMENET metal6 metal6 0.46  STACK ;
```

```
SAMENET cut1 cut1 0.26 ;
SAMENET cut2 cut2 0.26 ;
SAMENET cut3 cut3 0.26 ;
SAMENET cut4 cut4 0.26 ;
SAMENET cut5 cut5 0.35 ;
```

```
SAMENET cut1 cut2 0.0 STACK ;
SAMENET cut2 cut3 0.0 STACK ;
SAMENET cut3 cut4 0.0 STACK ;
SAMENET cut4 cut5 0.0 STACK ;
```

END SPACING

Calculating and Fixing Process Antenna Violations

This appendix describes process antenna violations and how you can use the router to fix them. It includes the following sections:

- [Overview](#) on page 192
- [Using Process Antenna Keywords in the LEF and DEF Files](#) on page 196
- [Calculating Antenna Ratios](#) on page 197
- [Checking for Antenna Violations](#) on page 212
- [Using Antenna Diode Cells](#) on page 217
- [Using DiffUseOnly](#) on page 218
- [Calculations for Hierarchical Designs](#) on page 219

Overview

During deep submicron wafer fabrication, gate damage can occur when excessive static charges accumulate and discharge, passing current through a gate. If the area of the layer connected directly to the gate or connected to the gate through lower layers is large relative to the area of the gate and the static charges are discharged through the gate, the discharge can damage the oxide that insulates the gate and cause the chip to fail. This phenomenon is called the *process antenna effect* (PAE).

To determine the extent of the PAE, the router calculates the area of the layer relative to the area of the gates connected to it, or connected to it through lower layers. The number it calculates is called the *antenna ratio*. Each foundry sets a maximum allowable antenna ratio for the chips it fabricates.

For example, assume a foundry sets a maximum allowable antenna ratio of 500. If a net has two input gates that each have an area of 1 square micron, any metal layers that connect to the gates and have an area larger than 1,000 square microns have process antenna violations because they would cause the antenna ratio to be higher than 500:

$$\text{Antenna Ratio} = \frac{\text{Area of metal layer}}{\text{Area of gates}} \quad 500 = \frac{1000}{1 + 1}$$

To tell the router the values to use when it calculates the antenna ratio, you set antenna keywords in the LEF and DEF files. The router measures potential damage caused by PAE by checking the ratio it calculates against the values specified by the antenna keywords. When it finds a net whose antenna ratio for a specified layer exceeds the maximum allowed value for that layer, it finds a *process antenna violation* and attempts to fix it using one or both of the following methods:

- Changing the routing so the routing layers connected to a gate or connected to a gate through lower layers are not so large that they build enough static charge to damage the gate
- Inserting diodes that protect the gate by providing an alternate path to discharge the static charge

LEF can specify several types of antenna ratios, including ratios for PAE damage on one layer only and ratios calculated by adding accumulated damage on several layers. In addition, LEF can specify ratios based on the area of the metal wires or the cut area of vias.

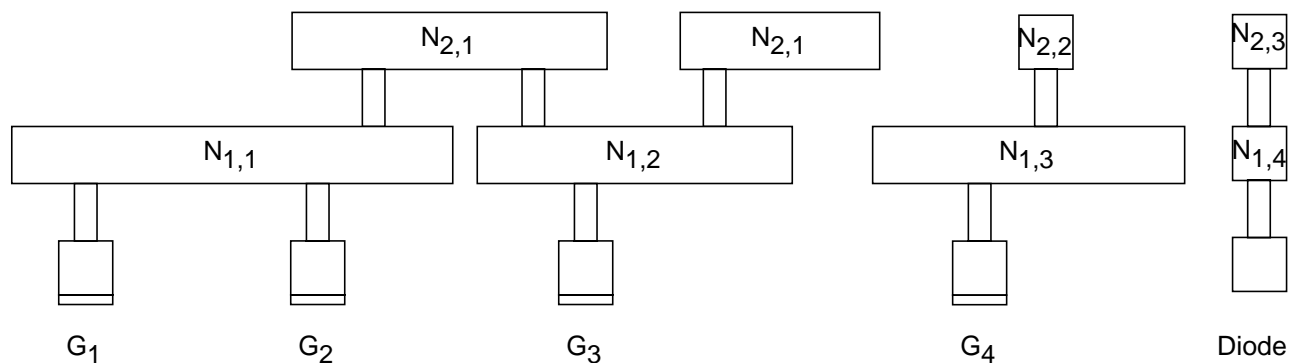
What Are Process Antennas?

In a chip manufacturing process, metal layers are built up, layer by layer, starting with the first-level metal layer (usually referred to as *metal1*). Next, the *metal1-metal2* vias are created, then the second-level metal layer, then *metal2-metal3* vias, and so on.

On each metal layer, metal is initially deposited so it covers the entire chip. Then, the unneeded portions of the metal are removed by etching, typically in plasma (charged particles).

Figure D-1 on page 193 shows a section of an imaginary chip after the unneeded metal from *metal2* is removed.

Figure D-1



In the figure,

- Gate areas for transistors are labelled G_k , where k is a sequential number starting with 1.
- Wire segments are labelled $N_{i,j}$
 - N signifies that the wire segment is an electrically connected node
 - i specifies the metal layer to which the node belongs
 - j is a sequential number for the node on that metal layer
- Nodes are labelled so that all pieces of the metal geometry on layer *metal_i* that are electrically connected by conductors at layers below *metal_i* belong to the same node. For example, the two *metal2* wire segments that belong to node $N_{2,1}$ are electrically connected to gates G_1 , G_2 , and G_3 by a piece of wire on *metal1* (labelled $N_{1,2}$).

Thick oxide insulates the already-fabricated structures below *metal2*, preventing them from direct contact with the plasma. The *metal2* geometries, however, are exposed to the plasma,

and collect charge from it. As the metal geometries collect charge, they build up voltage potential.

Because the metal geometries collect charge during the metallization process, they are referred to as process antennas. In general, the more area covered by the metal geometries that are exposed to the plasma (that is, the larger the process antennas), the more charge they can collect.

In the figure, note the following:

- Node $N_{1,1}$ is electrically connected to gates G_1 and G_2 .
- Node $N_{1,2}$ is electrically connected to gate G_3 .
- Node $N_{2,1}$ (node $N_{2,1}$ has two pieces of metal) is electrically connected to gates G_1 , G_2 , and G_3 .
- Node $N_{1,3}$ and node $N_{2,2}$ are electrically connected to gate G_4 .
- Node $N_{1,4}$ and node $N_{2,3}$ are electrically connected to the diffusion (diode).

What Is the Process Antenna Effect (PAE)?

If the voltage potential across the gate oxide becomes large enough to cause current to flow across the gate oxide, from the process antennas to the gates to which the process antennas are electrically connected, the current can damage the gate oxide. The process antenna effect (PAE) is the term used to describe the build-up of charge and increase in voltage potential. The larger the total gate area that is electrically connected to the process antennas on a specific layer, the more charge the connected gates can withstand.

In the imaginary chip in the figure, if the current were to flow, the following would happen, as a result of the node-gate connections:

- The charge collected by process antennas on nodes $N_{1,1}$, $N_{1,2}$, and $N_{2,1}$ would be discharged through one or more of gates G_1 , G_2 , and G_3 .
- The charge collected by process antennas on nodes $N_{1,3}$ and $N_{2,2}$ would be discharged through gate G_4 .
- The charge collected by process antennas on node $N_{1,4}$ and $N_{2,3}$ would be discharged through the diode.

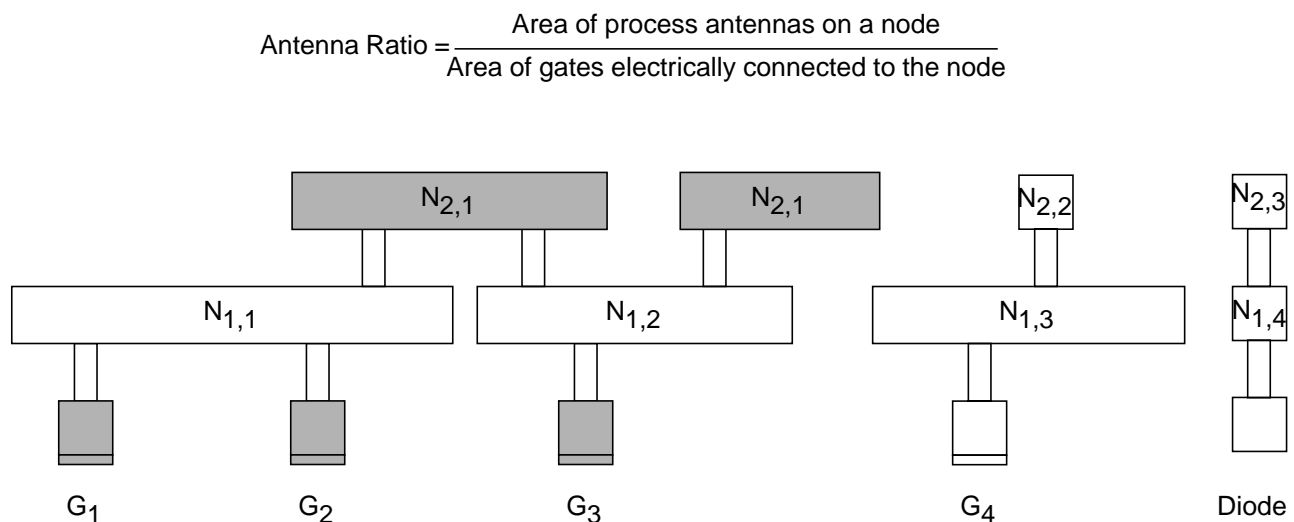
What Is the Antenna Ratio?

Because the total gate area that is electrically connected to a node (and therefore connected to the process antennas) determines the amount of charge from the process antennas the electrically connected gates can withstand, and because the size of the process antennas connected to the node determines how much charge the antennas collect, it is useful to calculate the ratio of the size of the process antennas on a node to the size of the gate area that is electrically connected to the node. This is the antenna ratio. The greater the antenna ratio, the greater the potential for damage to the gate oxide.

If you check a chip and obtain an antenna ratio greater than the threshold specified by the foundry, gate damage is likely to occur.

Figure D-2 on page 195 shows the same section of the imaginary chip as the previous figure. The shaded areas in this figure represent the process antennas on node $N_{2,1}$ and the gates to which they connect: gates G_1 , G_2 , and G_3 . The shaded gates discharge the electricity collected by the process antennas on node $N_{2,1}$.

Figure D-2



What Can Be Done to Improve the Antenna Ratio?

If there is an alternate path for the current to flow, the charge on the node can be discharged through the alternate path before the voltage potential reaches a level that damages the gate. For example, a Zenner diode, which allows current to flow in the reverse direction when the reverse bias reaches a specified breakdown voltage, provides an alternate path, and helps avoid building up so much charge at the node that the charge is discharged through the gate

oxide. Diffusion features that form the output of a logic gate (source and drain of transistors) can provide such an alternate discharge path.

Routers typically use two methods to decrease the antenna ratio:

- Changing the routing by breaking the metal layers into smaller pieces
- Inserting antenna diode cells to discharge the current

Both of these methods supply alternate paths for the current. For details about how to specify antenna diode cells, see [“Using Antenna Diode Cells”](#) on page 217.

Using Process Antenna Keywords in the LEF and DEF Files

You tell the router the values to use for the gate, diffusion, and metal areas by setting values for process antenna keywords in the LEF and DEF files for your design. You also tell the router the values to use for the threshold process antenna ratios by setting the keywords.

The following table lists LEF version 5.5 antenna keywords.

If the keyword ends with ...	It refers to ...	Examples
area	Area of the gates or diffusion Measured in square microns	ANTENNADIFFAREA ANTENNAGATEAREA
factor	Area multiplier used for the metal nodes	ANTENNAAREAFactor ANTENNASIDEAREAFactor

Note: Use `DIFFUSEONLY` if you want the multiplier to apply only when connecting to diffusion. For more information, see [“Using DiffUseOnly”](#) on page 218.

LEF/DEF 5.4 Language Reference

Calculating and Fixing Process Antenna Violations

If the keyword ends with ...	It refers to ...	Examples
ratio	Relationship the router is calculating Cum is used in keywords for cumulative antenna ratio.	ANTENNAAREARATIO ANTENNASIDEAREARATIO ANTENNADIFFAREARATIO ANTENNADIFFSIDEAREARATIO ANTENNACUMAREARATIO ANTENNACUMSIDEAREARATIO ANTENNACUMDIFFAREARATIO ANTENNACUMDIFFSIDEAREARATIO

Calculating Antenna Ratios

Tools should calculate antenna ratios using one of the following models:

■ The partial checking model

Using this model, you calculate damage to gates by process antennas on one layer. For example, if you use the partial checking model to calculate the PAE referred to a gate from *metal3*, you do not consider any potential damages referred to that gate from metallization steps on *metal1* or *metal2*.

You use this model to calculate a partial antenna ratio (PAR). A PAR tells you if any single metallization step is likely to inflict damage to a gate.

■ The cumulative checking model

This model is more conservative than the partial checking model. It adds damage to a gate caused by the PAE referred to the gate from each metallization step, starting from *metal1* up to the layer that is being checked. For example, if you use the cumulative checking model to calculate the PAE referred to a gate from *metal3*, you add the PAR from the relevant antenna areas on *metal1*, *metal2*, and *metal3*.

You use this model to calculate a cumulative antenna ratio (CAR). A CAR adds the damages on successive layers together to accumulate them as the layers are built up.

Calculating the Antenna Area

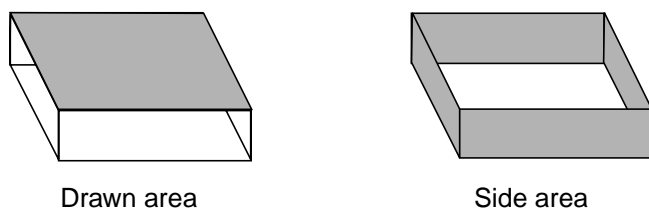
The area used to model the charge-collecting ability of a node is called the *antenna area*. The router calculates the antenna area for one of the following areas:

■ The drawn area (the top surface area of the process antenna)

- The side area (the area of the sides of the process antenna)

Figure D-3 on page 198 shows drawn and side areas.

Figure D-3



Antenna Area Factor

You can increase or decrease the calculated antenna area by specifying an antenna area factor in the LEF file.

- Use `ANTENNAAREAFactor` to adjust the calculation of the drawn area.
- Use `ANTENNASIDEAREAFactor` to adjust the calculation of the side area.

The default value of both factors is 1.

Calculating a PAR

The formula to calculate a PAR is

$$PAR(N_{i,j}, G_k) = \frac{Area(N_{i,j}) \times F}{\sum_{G_k \in C(N_{i,j})} Area(G_k)}$$

$PAR(N_{i,j}, G_k)$ is the partial antenna ratio for node j on *metal* _{i} with respect to gate G_k , where G_k is electrically connected to node $N_{i,j}$ by layer i or below.

$Area(N_{i,j})$ is the drawn or side area of node $N_{i,j}$.

F is an optional scaling factor (`AntennaAreaFactor` or `AntennaSideAreaFactor`).

LEF/DEF 5.4 Language Reference

Calculating and Fixing Process Antenna Violations

$C(N_{i,j})$ is the set of gates G_k that are electrically connected to $N_{i,j}$ through the layers below $metal_i$.

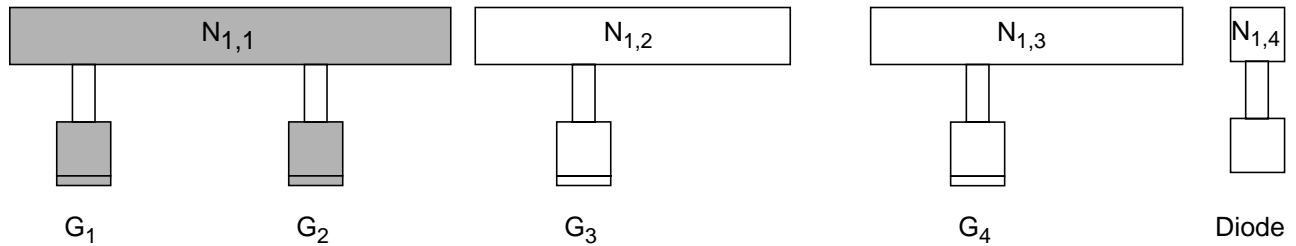
$Area(G_k)$ is the drawn or side area of gate G_k . (The reason to include the G_k parameter for PAR is to maintain uniformity with the notation for CAR.)

Note: For a specified node $N_{i,j}$, the $PAR(N_{i,j}, G_k)$ for all gates G_k that are connected to the node $N_{i,j}$ using $metal_i$ or below are identical.

Calculations for PAR on the First Metal Layer

Figure D-4 on page 199 shows a section of an imaginary chip after the first metal layer is processed.

Figure D-4



The shaded areas in the figure represent the wire segment and the gates whose areas you must compute to evaluate the formula below.

To calculate $PAR(N_{i,j}, G_k)$ for node $N_{1,1}$, a node on the first metal layer, with respect to gate G_1 , use the following formula:

$$PAR(N_{1,1}, G_1) = \frac{Area(N_{1,1}) \times F}{Area(G_1) + Area(G_2)}$$

Because gates G_1 and G_2 both connect to node $N_{1,1}$, the following statement is true:

$$PAR(N_{1,1}, G_1) = PAR(N_{1,1}, G_2)$$

To calculate PAR for node $N_{1,2}$, another node on the first metal layer, with respect to gate G_3 , use the following formula:

$$PAR(N_{1,2}, G_3) = \frac{Area(N_{1,2}) \times F}{Area(G_3)}$$

LEF/DEF 5.4 Language Reference

Calculating and Fixing Process Antenna Violations

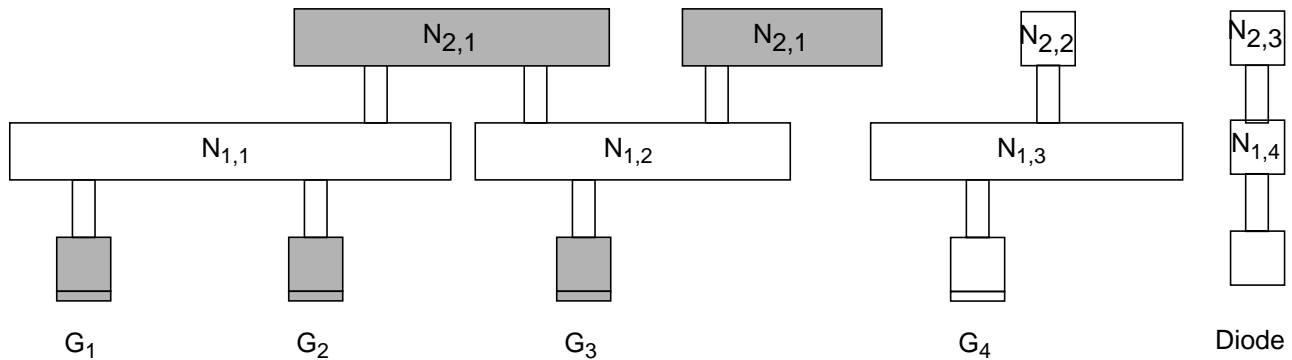
To calculate $PAR(N_{i,j}, G_k)$ for node $N_{1,3}$, another node on the first metal layer, with respect to gate G_4 , use the following formula:

$$PAR(N_{1,3}, G_4) = \frac{Area(N_{1,3}) \times F}{Area(G_4)}$$

Calculations for PAR on the Second Metal Layer

Figure D-5 on page 200 shows the chip after the second metal layer is processed.

Figure D-5



The shaded areas in the figure represent the wire segments and the gates whose areas you must compute to evaluate the formula below.

$N_{2,1}$ consists of two pieces of metal on the second layer that are electrically connected at this step in the fabrication process. Therefore, to calculate $PAR(N_{2,1}, G_1)$, you must add the area of both pieces together.

To calculate $PAR(N_{i,j}, G_k)$ for node $N_{2,1}$, a node on the second metal layer, with respect to gate G_1 , use the following formula:

$$PAR(N_{2,1}, G_1) = \frac{Area(N_{2,1}) \times F}{Area(G_1) + Area(G_2) + Area(G_3)}$$

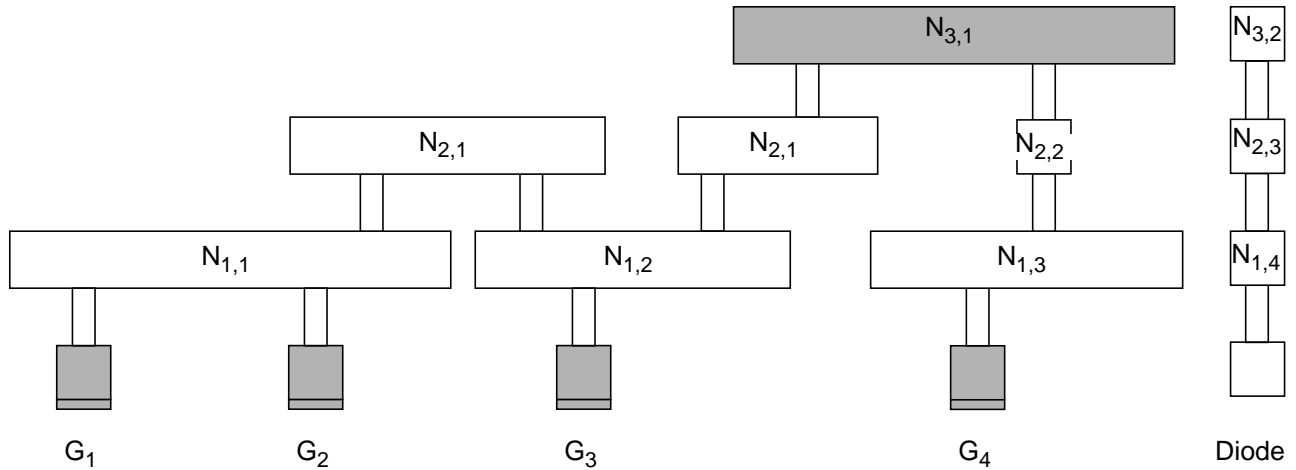
As on the first layer,

$$PAR(N_{2,1}, G_1) = PAR(N_{2,1}, G_2) = PAR(N_{2,1}, G_3)$$

Calculations for PAR on the Third Metal Layer

Figure D-6 on page 201 shows the chip after the third metal layer is processed.

Figure D-6



The shaded areas in the figure represent the wire segment and the gates whose areas you must compute to evaluate the formula below.

To calculate $PAR(N_{i,j}, G_k)$ for node $N_{3,1}$, a node on the third metal layer, with respect to gate G_1 , use the following formula:

$$PAR(N_{3,1}, G_1) = \frac{Area(N_{3,1}) \times F}{Area(G_1) + Area(G_2) + Area(G_3) + Area(G_4)}$$

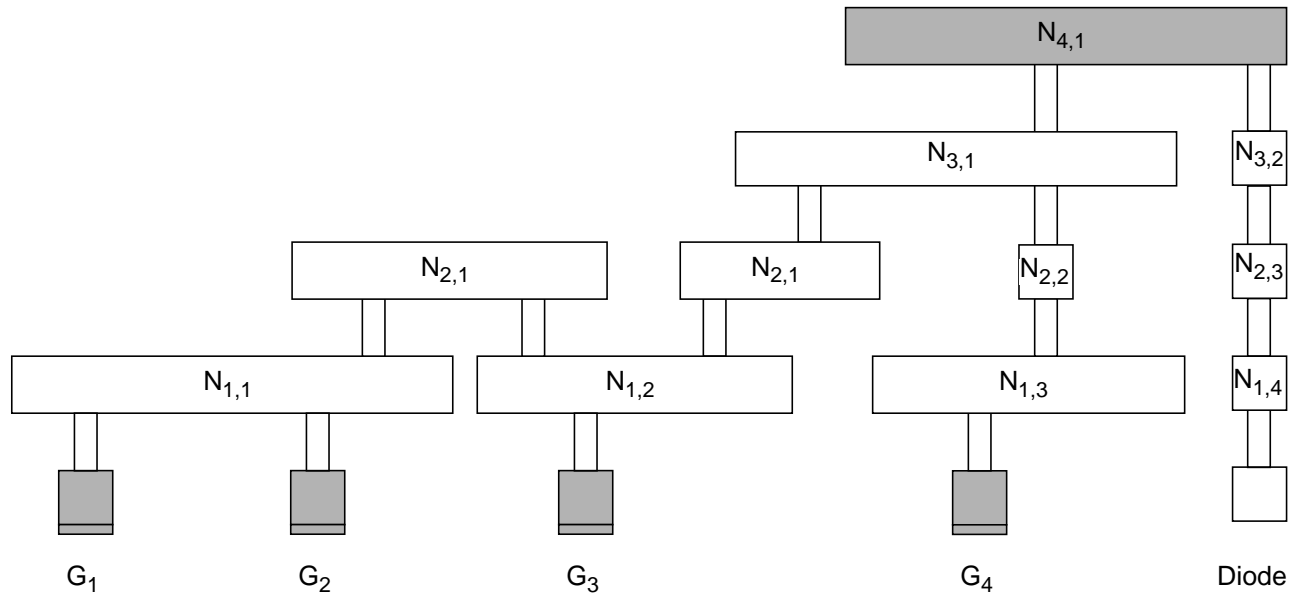
As on the prior layers,

$$PAR(N_{3,1}, G_1) = PAR(N_{3,1}, G_2) = PAR(N_{3,1}, G_3) = PAR(N_{3,1}, G_4)$$

Calculations for PAR on the Fourth Metal Layer

Figure D-7 on page 202 shows the chip after the fourth metal layer is processed.

Figure D-7



The shaded areas in the figure represent the wire segment and the gates whose areas you must compute to evaluate the formula below.

To calculate $PAR(N_{i,j}, G_k)$ for the fourth metal layer, use the following formula:

$$PAR(N_{4,1}, G_1) = \frac{Area(N_{4,1}) \times F}{Area(G_1) + Area(G_2) + Area(G_3) + Area(G_4)}$$

As on the prior layers,

$$PAR(N_{4,1}, G_1) = PAR(N_{4,1}, G_2) = PAR(N_{4,1}, G_3) = PAR(N_{4,1}, G_4)$$

Note: Node $N_{4,1}$ is connected to the diffusion layer through the output diode. After the router calculates the antenna ratio, it compares its calculations to the area of the diffusion, instead of the area of the gates.

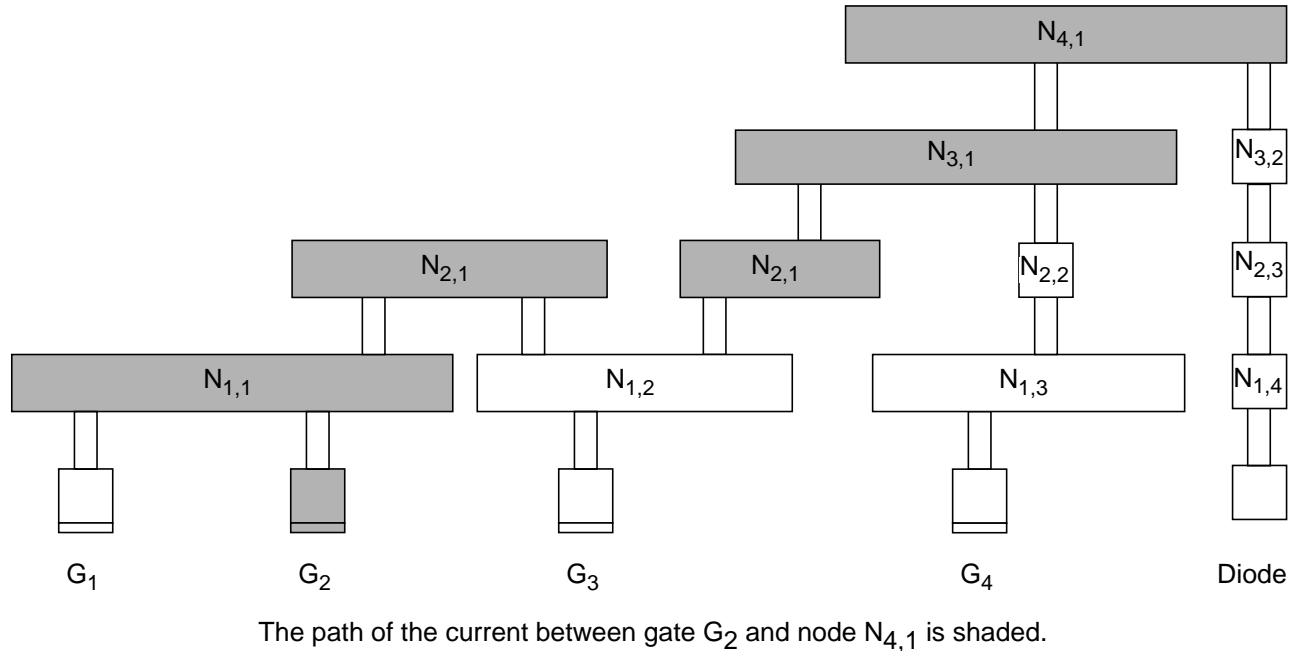
Calculating a CAR

To calculate a CAR, the router adds the PARs for all the relevant nodes on the specified or lower metal layers that are electrically connected to a gate. Therefore, $CAR(N_{i,j}, G_k)$ designates the cumulative damage to gate G_k by metallization steps up to the current level of metal, i .

Note: In practice, the router only needs to keep track of the worst-case CAR; however the CARs for all of the gates shown in [Figure D-8](#) on page 203 are described here.

The router calculates an antenna ratio with respect to a node-gate pair. To find the CAR for the node $N_{i,j}$ - gate G_k pair, you trace the path of the current between gate G_k and node $N_{i,j}$ and add the PAR with respect to gate G_k for the all nodes in the path between the first metal layer and layer i that you can trace back to G_k .

Figure D-8



Important

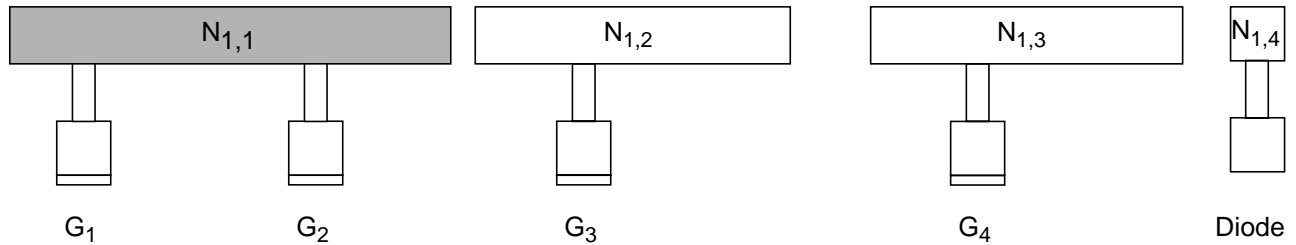
In [Figure D-8](#) on page 203, node $N_{1,2}$ is not shaded because it was not electrically connected to G_2 when *metal1* was processed. That is, because the charge accumulated on $N_{1,2}$ when *metal1* was processed cannot damage gate G_1 , the router does not include it in the calculations for $CAR(N_{2,1}, G_1)$.

Another way to explain this is to say that the PAE from node $N_{1,2}$ with respect to gate G_2 is 0.

Calculations for CAR on the First Metal Layer

[Figure D-9](#) on page 204 shows the chip after the first metal layer is processed.

Figure D-9



In the figure above,

$$\text{CAR}(N_{1,1}, G_1) = \text{PAR}(N_{1,1}, G_1)$$

$$\text{CAR}(N_{1,1}, G_2) = \text{PAR}(N_{1,1}, G_2)$$

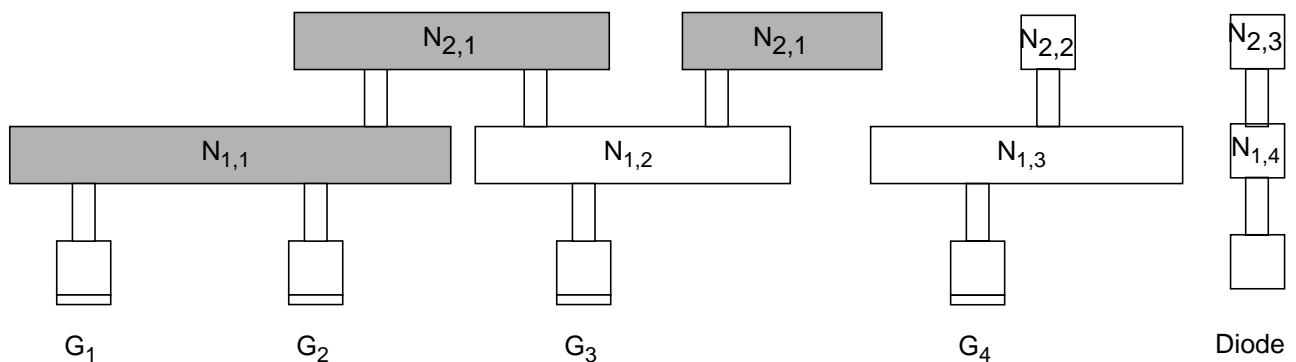
Because $\text{PAR}(N_{1,1}, G_1)$ equals $\text{PAR}(N_{1,1}, G_2)$, $\text{CAR}(N_{1,1}, G_1)$ equals $\text{CAR}(N_{1,1}, G_2)$.

Note: In general, $\text{CAR}(N_{i,j}, G_k)$ equals $\text{CAR}(N_{i,j}, G_{k'})$ if the two gates G_k and $G_{k'}$ are electrically connected to the same node on *metal1*, the lowest layer that is subject to the process antenna effect.

Calculations for CAR on the Second Metal Layer

Figure D-10 on page 204 shows the chip after the second metal layer is processed.

Figure D-10



The path of the current between gate G_1 and node $N_{2,1}$ is shaded.
Note that node $N_{2,1}$ comprises two pieces of metal.

LEF/DEF 5.4 Language Reference

Calculating and Fixing Process Antenna Violations

Important

In the figure above, $N_{1,2}$ is not included in the calculations for $CAR(N_{2,1}, G_1)$ because it was not electrically connected to G_1 when *metal1* was processed. That is, because the charge accumulated on $N_{1,2}$ when *metal1* was processed cannot damage gate G_1 , the router does not include it in the calculations for $CAR(N_{2,1}, G_1)$.

In the figure above,

$$CAR(N_{2,1}, G_1) = PAR(N_{1,1}, G_1) + PAR(N_{2,1}, G_1)$$

$$CAR(N_{2,1}, G_2) = PAR(N_{1,1}, G_2) + PAR(N_{2,1}, G_2)$$

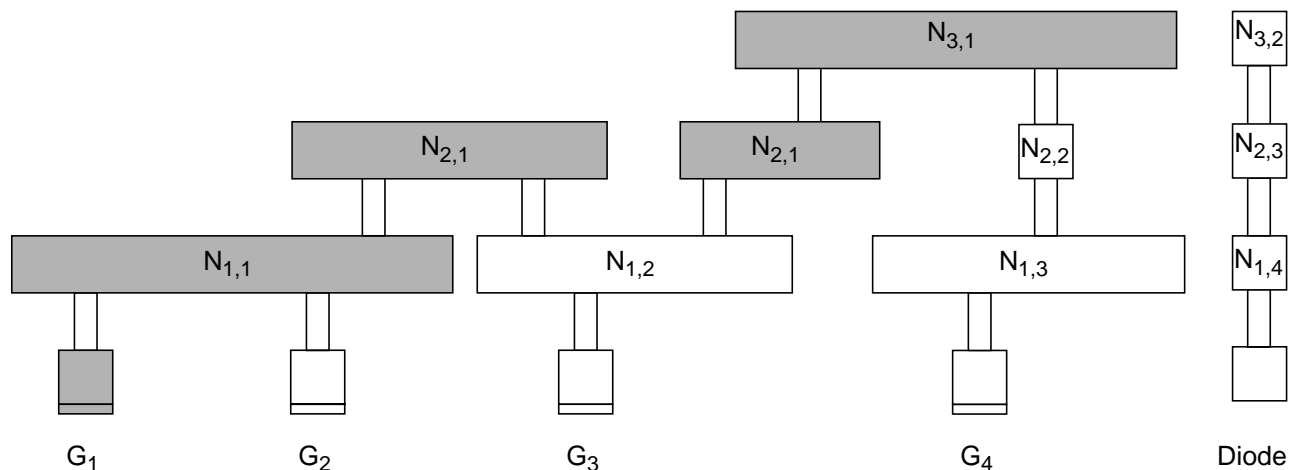
Gates G_1 and G_2 have the same history with regard to PAE because they are connected to the same piece of *metal1*, so they have the same CAR for any node on a specified layer:

$$CAR(N_{2,1}, G_1) = CAR(N_{2,1}, G_2)$$

Calculations for CAR on the Third Metal Layer

Figure D-11 on page 205 shows the chip after the third metal layer is processed.

Figure D-11



Gate G_1

In the figure above,

$$\text{CAR}(N_{3,1}, G_1) = \text{PAR}(N_{1,1}, G_1) + \text{PAR}(N_{2,1}, G_1) + \text{PAR}(N_{3,1}, G_1)$$

Gate G_2

In the figure above,

$$\text{CAR}(N_{3,1}, G_2) = \text{PAR}(N_{1,1}, G_2) + \text{PAR}(N_{2,1}, G_2) + \text{PAR}(N_{3,1}, G_2)$$

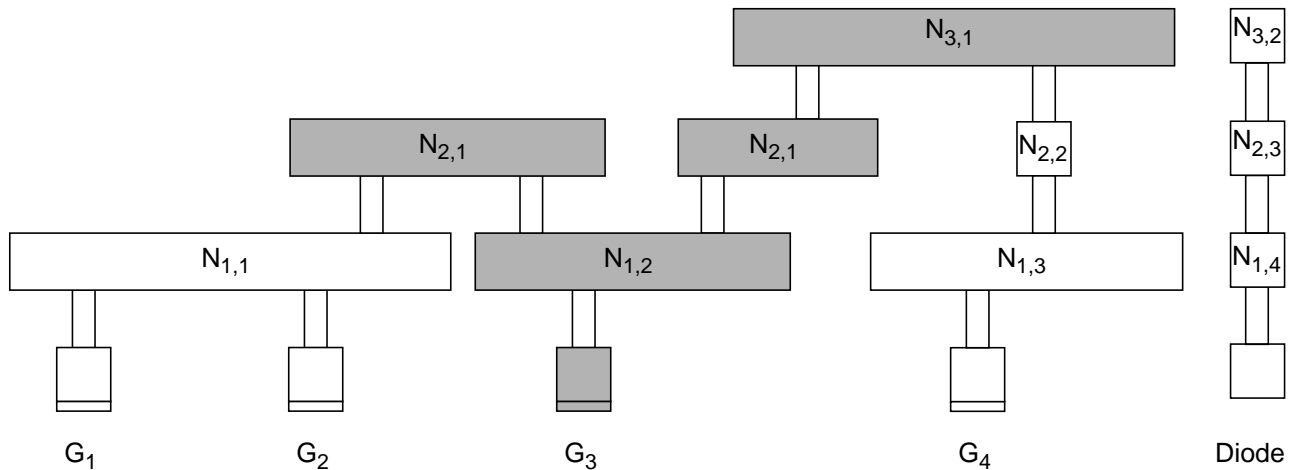
$\text{CAR}(N_{3,1}, G_1)$ equals $\text{CAR}(N_{3,1}, G_2)$ because gates G_1 and G_2 are both electrically connected to the same node, $N_{1,1}$, on *metal1* and therefore have the same history with regard to PAE. Therefore, the formula for $\text{CAR}(N_{3,2}, G_2)$ is $\text{CAR}(N_{3,1}, G_1) = \text{CAR}(N_{3,1}, G_2)$

Gates G_3 and G_4

Gates G_3 and G_4 are not connected to the same node on *metal1* and therefore do not have the same history with regard to PAE. Therefore, the $\text{CAR}(N_{3,1}, G_3)$ and $\text{CAR}(N_{3,1}, G_4)$ do not necessarily equal $\text{CAR}(N_{3,1}, G_1)$ or $\text{CAR}(N_{3,1}, G_2)$.

In [Figure D-12](#) on page 206, the relevant areas for calculating CAR for gate G_3 are shaded.

Figure D-12



In the figure above,

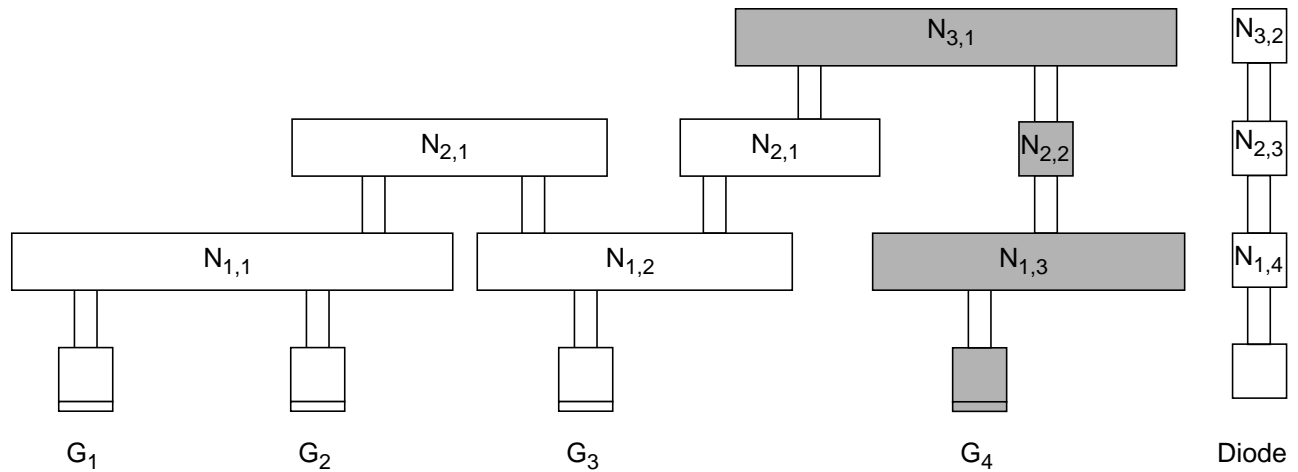
LEF/DEF 5.4 Language Reference

Calculating and Fixing Process Antenna Violations

$$\text{CAR}(N_{3,1}, G_3) = \text{PAR}(N_{1,2}, G_3) + \text{PAR}(N_{2,1}, G_3) + \text{PAR}(N_{3,1}, G_3)$$

In [Figure D-13](#) on page 207, the relevant areas for calculating CAR for gate G_4 are shaded.

Figure D-13



In the figure above,

$$\text{CAR}(N_{3,1}, G_4) = \text{PAR}(N_{1,3}, G_4) + \text{PAR}(N_{2,2}, G_4) + \text{PAR}(N_{3,1}, G_4)$$

Calculations for CAR on the Fourth Metal Layer

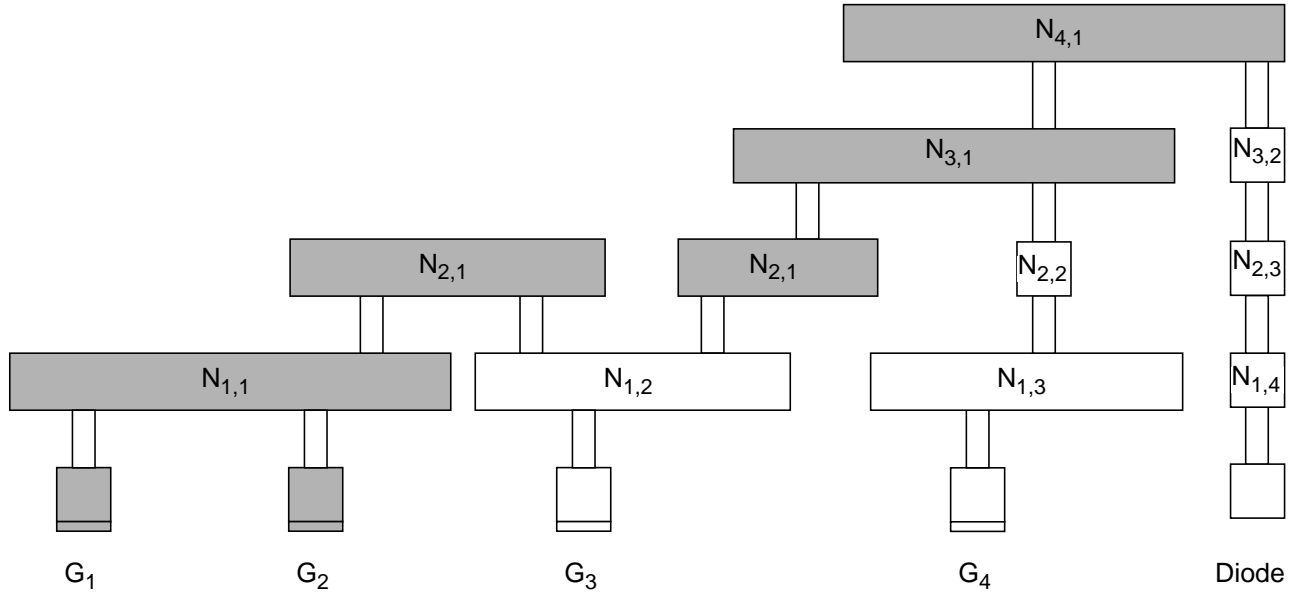
The following figure shows the chip after the fourth metal layer is processed.

Note: Node $N_{4,1}$ is connected to the diffusion layer through the output diode. After the router calculates the antenna ratio, it compares its calculations to the area of the diffusion, instead of the area of the gates.

Gates G_1 and G_2

In [Figure D-14](#) on page 208, the relevant areas for calculating $\text{CAR}(N_{4,1}, G_1)$ and $\text{CAR}(N_{4,1}, G_2)$ are shaded.

Figure D-14



In the figure above,

$$\text{CAR}(N_{4,1}, G_1) = \text{PAR}(N_{1,1}, G_1) + \text{PAR}(N_{2,1}, G_1) \\ + \text{PAR}(N_{3,1}, G_1) + \text{PAR}(N_{4,1}, G_1)$$

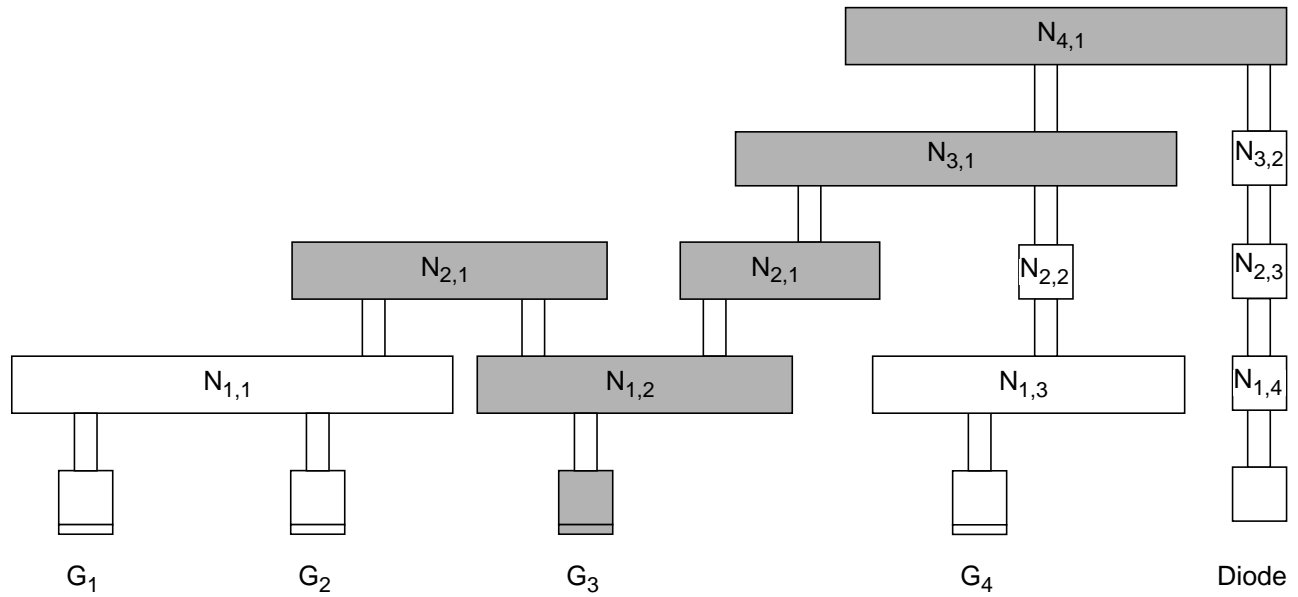
$$\text{CAR}(N_{4,1}, G_2) = \text{PAR}(N_{1,1}, G_2) + \text{PAR}(N_{2,1}, G_2) \\ + \text{PAR}(N_{3,1}, G_2) + \text{PAR}(N_{4,1}, G_2)$$

$$\text{CAR}(N_{4,1}, G_1) = \text{CAR}(N_{4,1}, G_2)$$

Gate G_3

In [Figure D-15](#) on page 209, the relevant areas for calculating $\text{CAR}(N_{4,1}, G_3)$ are shaded.

Figure D-15



In the figure above,

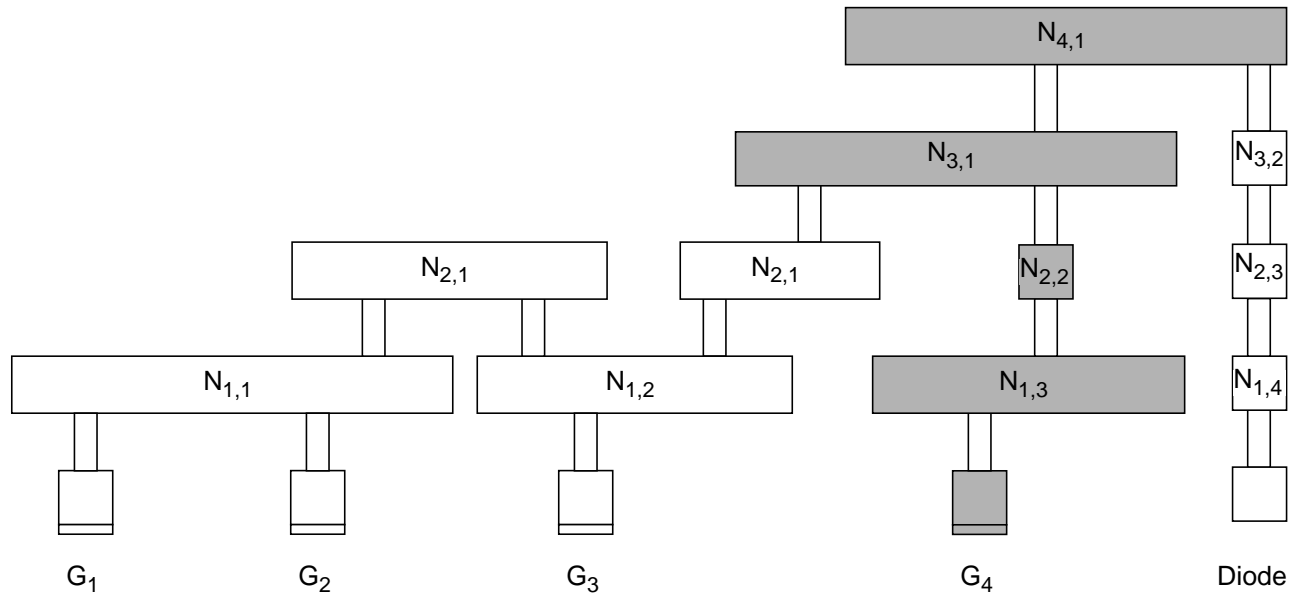
$$\begin{aligned} \text{CAR}(N_{4,1}, G_3) &= \text{PAR}(N_{1,2}, G_3) + \text{PAR}(N_{2,1}, G_3) \\ &+ \text{PAR}(N_{3,1}, G_3) + \text{PAR}(N_{4,1}, G_3) \end{aligned}$$

$\text{CAR}(N_{4,1}, G_3)$ does not equal $\text{CAR}(N_{4,1}, G_1)$ or $\text{CAR}(N_{4,1}, G_2)$ because it is not connected to the same node on *metal1*.

Gate G_4

In [Figure D-16](#) on page 210, the relevant areas for calculating $\text{CAR}(N_{4,1}, G_4)$ are shaded.

Figure D-16



In the figure above,

$$\begin{aligned} \text{CAR}(N_{4,1}, G_4) = & \text{PAR}(N_{1,3}, G_4) + \text{PAR}(N_{2,2}, G_4) \\ & + \text{PAR}(N_{3,1}, G_4) + \text{PAR}(N_{4,1}, G_4) \end{aligned}$$

$\text{CAR}(N_{4,1}, G_4)$ does not equal $\text{CAR}(N_{4,1}, G_1)$, $\text{CAR}(N_{4,1}, G_2)$, or $\text{CAR}(N_{4,1}, G_3)$ because it is not connected to the same node on *metal1*.

Calculating Ratios for a Cut Layer

The router calculates damage from a cut layer separately from damage from a metal layer. Calculations for the cut layers do not use side area modelling.

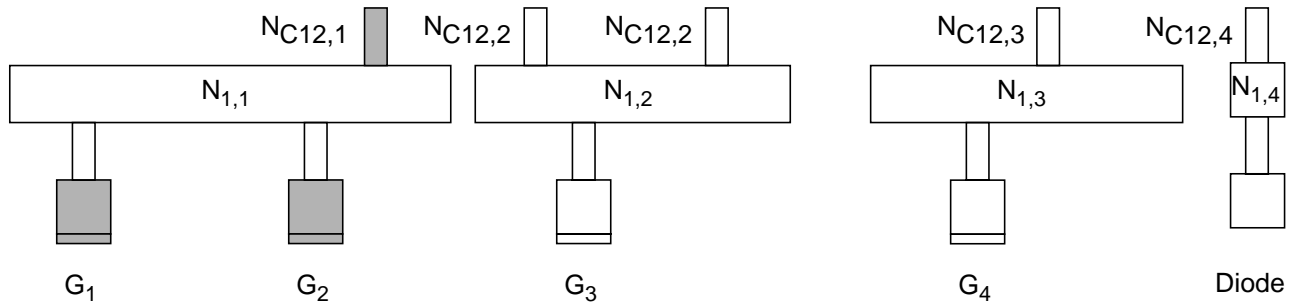
In the figures and text that follow,

- C_{ij} is the cut layer between *metal_i* and *metal_j*.
- $N_{C_{ij},k}$ specifies an electrically connected node on C_{ij} .
- The nodes are numbered sequentially, from left to right.

Calculating a PAR on a Cut Layer

[Figure D-17](#) on page 211 shows the chip after the C12 process step.

Figure D-17



In the figure above,

$$PAR(N_{C12,1}, G_1) = \frac{Area(N_{C12,1}, G_1) \times F}{Area(G_1) + Area(G_2)}$$

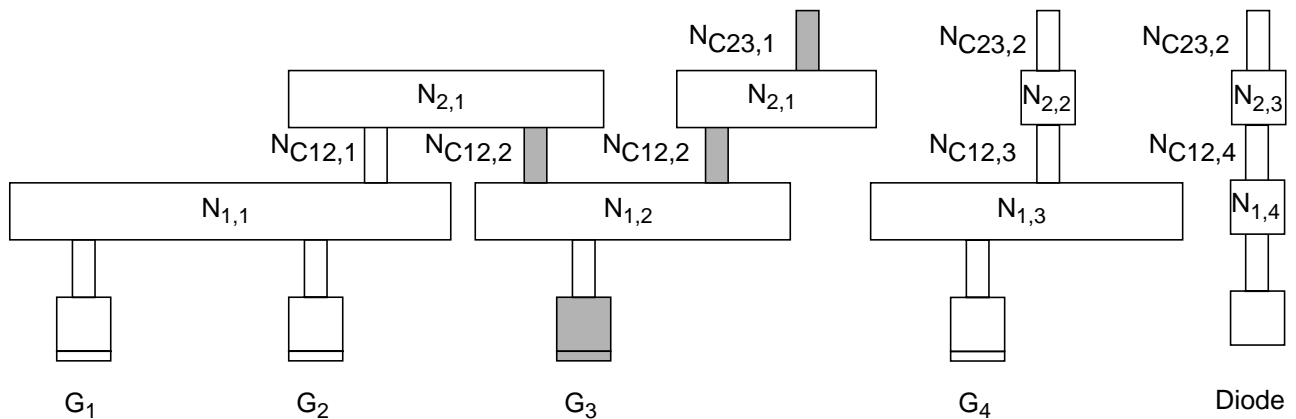
As in calculations on the metal layers,

$$PAR(N_{C12,1}, G_1) = PAR(N_{C12,2}, G_2)$$

Calculating a CAR on a Cut Layer

Figure D-18 on page 211 shows the chip after the C23 process step.

Figure D-18

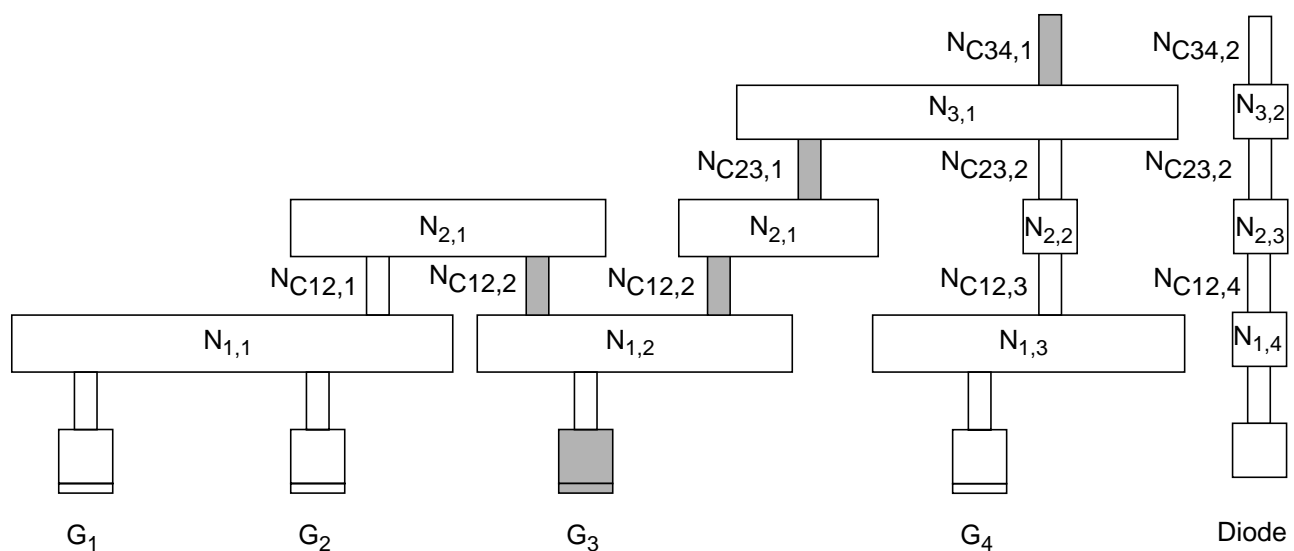


The router calculates the CAR with respect to gate G_3 after the cut C23 process step as follows:

$$CAR(N_{C23,1},G_3) = \frac{Area(N_{C23,1},G_3) \times F}{Area(G_3)} + \frac{Area(N_{C12,2},G_3) \times F}{Area(G_3)}$$

Figure D-19 on page 212 shows the chip after the C34 process step.

Figure D-19



The router calculates the CAR with respect to gate G_3 after the cut C34 process step as follows:

$$CAR(N_{C34,1,G_3}) = \frac{Area(N_{C34,1,G_3}) \times F}{Area(G_3)} + \frac{Area(N_{C23,1,G_3}) \times F}{Area(G_3)} + \frac{Area(N_{C12,2,G_3}) \times F}{Area(G_3)}$$

Checking for Antenna Violations

For each metal layer, the router performs several antenna checks, using the keywords and values specified in the LEF or DEF file. The router can perform the following four types of antenna checks, depending on the keywords you set in the LEF file:

- Area Ratio Check
- Side Area Ratio Check

LEF/DEF 5.4 Language Reference

Calculating and Fixing Process Antenna Violations

- Cumulative Area Ratio Check
- Cumulative Side Area Ratio Check

Area Ratio Check

The area ratio check compares the PAR for each layer to the value of the `ANTENNAAREARATIO` or `ANTENNADIFFAREARATIO`.

The router calculates the PAR as follows:

$$\text{PAR}(N_{i,j}, G_k) = \frac{\text{Drawn area of } N_{i,j} \times \text{AntennaAreaFactor}}{\Sigma \text{ Area of gates connected below } N_{i,j}}$$

Note: The `ANTENNAAREAFactor` has a default value of 1.

According to the formula above, the area ratio check finds the PAR for node $N_{i,j}$ with respect to gate G_k by dividing the drawn area of the node by the area of the gates that are electrically connected to it. If the PAR is greater than the `ANTENNAAREARATIO` or `ANTENNADIFFAREARATIO` specified in the LEF file, the router finds a process antenna violation and attempts to fix it.

The link between $\text{PAR}(N_{i,j}, G_k)$ and a PAE violation at node $N_{i,j}$ depends on whether node $N_{i,j}$ is connected to a piece of diffusion, as follows:

- If there is no connection from node $N_{i,j}$ to a diffusion area through the current and lower layers, a violation occurs when the PAR is greater than the `ANTENNAAREARATIO`.
- If there is a connection from node $N_{i,j}$ to a diffusion area through current and lower layers, a violation occurs when the PAR is greater than the `ANTENNADIFFAREARATIO`.
- If there is a connection from node $N_{i,j}$ to a diffusion area through current and lower layers, and `ANTENNADIFFAREA` is not specified for an output or inout pin, the value is 0.

Side Area Ratio Check

The side area ratio check compares the PAR computed based on the side area of the nodes for each layer to the value of the `ANTENNASIDEAREARATIO` or `ANTENNADIFFSIDEAREARATIO`.

LEF/DEF 5.4 Language Reference

Calculating and Fixing Process Antenna Violations

The router calculates the PAR as follows:

$$\text{PAR}(N_{i,j}, G_k) = \frac{\text{Side area of } N_{i,j} \times \text{AntennaSideAreaFactor}}{\sum \text{Area of gates connected below } N_{i,j}}$$

Note: The ANTENNASIDEAREAFactor has a default value of 1.

According to the formula above, the area ratio check finds the PAR for node $N_{i,j}$ with respect to gate G_k by dividing the side area of the node by the area of the gates that are electrically connected to $N_{i,j}$. If the PAR is greater than the ANTENNASIDEAREARATIO or ANTENNADIFFSIDEAREARATIO specified in the LEF file, the router finds a process antenna violation and attempts to fix it.

The link between $\text{PAR}(N_{i,j}, G_k)$ and a PAE violation at node $N_{i,j}$ depends on whether node $N_{i,j}$ is connected to a piece of diffusion, as follows:

- If there is no connection to the diffusion area through the current and lower layers, a violation occurs when the PAR is greater than the ANTENNASIDEAREARATIO.
- If there is a connection to the diffusion area through current and lower layers, a violation occurs when the PAR is greater than the ANTENNADIFFSIDEAREARATIO.
- If there is a connection to the diffusion area through current and lower layers, and ANTENNADIFFAREA is not specified for an output or inout pin, the value is 0.

Cumulative Area Ratio Check

The cumulative area ratio check compares the CAR to the value of ANTENNACUMAREARATIO or ANTENNACUMDIFFAREARATIO. The CAR is equal to the sum of the PARs of all nodes on the same or lower layers that are electrically connected to the gate.

Note: When you use CARs, you can ignore metal layers by not specifying the CAR keywords for those layers. For example, if you want to check *metal1* using a PAR and the remaining metal layers using a CAR, you can define ANTENNAAREARATIO or ANTENNASIDEAREARATIO for *metal1*, and ANTENNACUMAREARATIO or ANTENNACUMSIDEAREARATIO for the remaining metal layers.

The cumulative area ratio check finds the CAR for node $N_{i,j}$ with respect to gate G_k by adding the PARs for all layers of metal, from the current layer down to *metal1*, for all nodes that are electrically connected G_k . If the CAR is greater than the ANTENNACUMAREARATIO or ANTENNACUMDIFFAREARATIO specified in the LEF file, the router finds a process antenna violation and attempts to fix it.

LEF/DEF 5.4 Language Reference

Calculating and Fixing Process Antenna Violations

The link between $CAR(N_{i,j}, G_k)$ and a PAE violation at node $N_{i,j}$ depends on whether node $N_{i,j}$ is connected to a piece of diffusion, as follows:

- If there is no connection to a diffusion area through the current and lower layers, a violation occurs when the CAR is greater than the `ANTENNACUMAREARATIO`.
- If there is a connection to a diffusion area through current and lower layers, a violation occurs when the CAR is greater than the `ANTENNACUMDIFFAREARATIO`.
- If there is a connection to a diffusion area through current and lower layers, and `ANTENNADIFFAREA` is not specified for an output or inout pin, the value is 0.

Cumulative Side Area Ratio Check

The cumulative side area ratio check compares the CAR to the value of the `ANTENNACUMSIDEAREARATIO` or `ANTENNACUMDIFFAREARATIO`.

Note: When you use CARs, you can ignore metal layers by not specifying the CAR keywords for those layers. For example, if you want to check *metal1* using a PAR and the remaining metal layers using a CAR, you can define `ANTENNAAREARATIO` or `ANTENNASIDEAREARATIO` for *metal1*, and `ANTENNACUMAREARATIO` or `ANTENNACUMSIDEAREARATIO` for the remaining metal layers.

The formula for this check is the same as for the `ANTENNACUMAREARATIO` check, except that the router uses the `ANTENNASIDEAREAFACOR` instead of the `ANTENNAAREAFACOR` in the calculations.

The cumulative side area ratio check finds the CAR for node $N_{i,j}$ with respect to gate G_k by adding the PARs for all layers of metal, from the current layer down to *metal1*, for all nodes that are electrically connected G_k . If the CAR is greater than the `ANTENNACUMSIDEAREARATIO` or `ANTENNACUMDIFFAREARATIO` specified in the LEF file, the router finds a process antenna violation and attempts to fix it.

- If there is no connection to a diffusion area through the current and lower layers, a violation occurs when the CAR is greater than the `ANTENNACUMSIDEAREARATIO`.
- If there is a connection to a diffusion area through current and lower layers, a violation occurs when the CAR is greater than the `ANTENNACUMSIDEAREARATIO`.
- If there is a connection to a diffusion area through current and lower layers, and `ANTENNACUMDIFFAREA` is not specified for an output or inout pin, the value is 0.

Example Using the Antenna Keywords

The following example is a portion of a LEF file that shows the antenna keywords for a process that has cumulative area ratio damage for metal and cut layers.

Assume you have the following antenna rules for your process:

1. A maximum cumulative metal to gate area ratio of 1000
2. If a diode of greater than .1 microns is connected to the metal, the maximum metal ratio is: $\text{ratio} = \text{diode_area} \times 2000 + 5000$
3. A maximum cumulative via to gate area ratio of 20
4. If a diode of greater than .1 microns is connected to the via, the maximum via ratio is: $\text{ratio} = \text{diode_area} \times 200 + 100$

The corresponding LEF file would include:

```
LAYER M1
  TYPE ROUTING ;
  ...
  ANTENNACUMAREARATIO 1000 ;
  ANTENNACUMDIFFAREARATIO
    PWL ( ( 0 1000 ) ( 0.099 1000 ) ( 0.1 5200 ) ( 100 205000 ) ) ;
END M1
```

```
LAYER VIA1
  TYPE CUT ;
  ...
  ANTENNACUMAREARATIO 20 ;
  ANTENNACUMDIFFAREARATIO
    PWL ( ( 0 20 ) ( 0.099 20 ) ( 0.1 120 ) ( 100 20100 ) ) ;
END VIA1
```

A typical standard cell that has only *M1* pins and routing inside of it would have:

```
MACRO INV1X
  CLASS CORE ;
  ...
  PIN IN
    DIRECTION INPUT ;
    ANTENNAGATEAREA .5 LAYER M1 ; # connects to 0.5  $\mu\text{m}^2$  poly gate
    ANTENNAPARTIALMETALAREA 1.0 LAYER M1 ; # has 1.0  $\mu\text{m}^2$  M1 area.
    # Note that it should not include the M1 pin area, just the M1 routing
```


LEF/DEF 5.4 Language Reference

Calculating and Fixing Process Antenna Violations

```
# area that is not included in the PIN shapes. In many cases, all of the
# M1 routing is included in the PIN, so this value is 0, and not in the
# LEF at all.
ANTENNAMAXAREACAR 10.0 LAYER M1 ; # has 10.0 cumulative ratio so far.
# This value can include area from internal poly routing if poly routing
# damage is accumulated with the metal layers. It does not include
# the area of the M1 pin area, just the M1 routing area that is not
# included in the PIN shapes. If poly damage is not included, and all
# of the M1 routing is included in the PIN, this value will be 0, and
# not in the LEF at all.
...
END IN
PIN OUT
    DIRECTION OUTPUT ;
    ANTENNADIFFAREA .2 LAYER M1 ; # connects to 0.2  $\mu\text{m}^2$  diffusion area
    ANTENNAPARTIALMETALAREA 1.0 LAYER M1 ; # has 1.0  $\mu\text{m}^2$  M1 area
    # No ANTENNAMAXAREACAR value because no internal poly gate is connected
    ...
END OUT
END INV1X
```

Using Antenna Diode Cells

Routers generally use one of two methods to fix process antenna violations:

- Change the routing by breaking the metal layers into smaller pieces
- Insert antenna diode cells to discharge the current

Changing the Routing

One method routers use to fix antenna violations is to limit the charge that is collected through the metal nodes exposed to the plasma. To do this, it goes up one layer or pushes the routing down one layer whenever the process antenna ratio exceeds the ratio set in the LEF file.

The router changes the routing by disconnecting nets with antenna violations and making the connections to higher metal layers instead. It does not make the connections to lower layers. This method works because the top metal layer always completes the connection from the gate to the output drain area of the driver, which is a diode that provides a discharge path.

Inserting Antenna Diode Cells

The second method routers use to repair antenna violations is to insert antenna diode cells in the design. The electrical charges on the metal that connects to the diodes is then discharged through the diode diffusion layer and substrate. The router inserts the diode cells automatically.

The following example shows a LEF definition of an antenna diode cell, with the CLASS CORE ANTENNACELL and ANTENNADIFFAREA defined:

```
MACRO antennal
  CLASS CORE ANTENNACELL ;
  ...
  PIN ANT1
    AntennaDiffArea 1.0 ;
    PORT
      LAYER metall ;
      RECT 0.190 2.380 0.470 2.660 ;
    END
  END ANT1
END antennal
```

Using DiffUseOnly

LEF defines only one value for ANTENNAAREAFactor and one value for ANTENNASIDEAREAFactor, with or without DIFFUSEONLY, per layer. If you specify more than one antenna area or side area factor for a layer, only the last one is used. The AREAFactor value lets you scale the value of the metal area. If you use the DIFFUSEONLY keyword, only metal attached to diffusion is scaled.

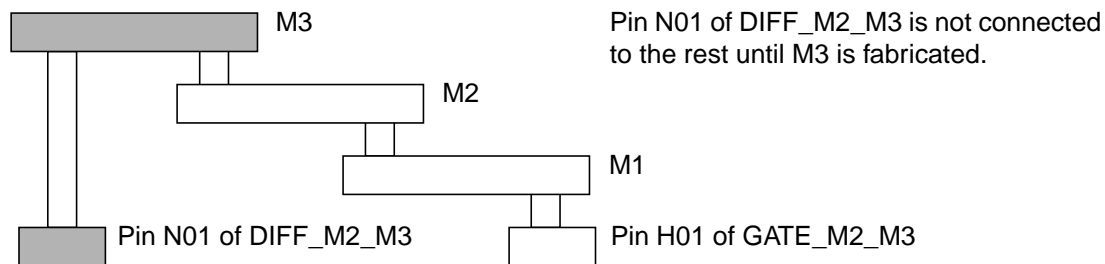
LEF/DEF 5.4 Language Reference

Calculating and Fixing Process Antenna Violations

Suppose you have the following LEF file:

```
Antenna.lef
-----
LAYER M3
TYPE ROUTING ;
PITCH 0.56 ;
DIRECTION HORIZONTAL ;
WIDTH 0.28 ; SPACING 0.28 ;
SPACING 0.36 RANGE 1.0 250.0 ;
CAPACITANCE CPERSQDIST 0.0009762 ;
RESISTANCE RPERSQ 0.129 ;
THICKNESS 0.60 ;
AntennaAreaRatio 10000 ;
AntennaDiffAreaRatio 10000 ;
AntennaAreaFactor 1.2 DiffUseOnly ;
AntennaSideAreaRatio 5000 ;
AntennaDiffSideAreaRatio 5000 ;
AntennaSideAreaFactor 1.4 DiffUseOnly ;
END M3
```

Figure D-20



In the figure,

- The input pin H01 of GATE_M2_M3 connects the metal wires to *metal1*, *metal2*, and *metal3* in sequence.
- The ANTENNAAREAFACTOR 1.2 DIFFUSEONLY and ANTENNASIDEAREAFACTOR 1.4 DIFFUSEONLY apply to *metal3* routing.
- Prior to *metal3* fabrication, there is no path to the diffusion diode. This causes the default factor of 1.0 to apply to the *metal1* and *metal2* segments shown when calculating PARs.

Calculations for Hierarchical Designs

The following section illustrates computation of antenna ratios for hierarchical designs.

LEF and DEF Keywords for Hierarchical Designs

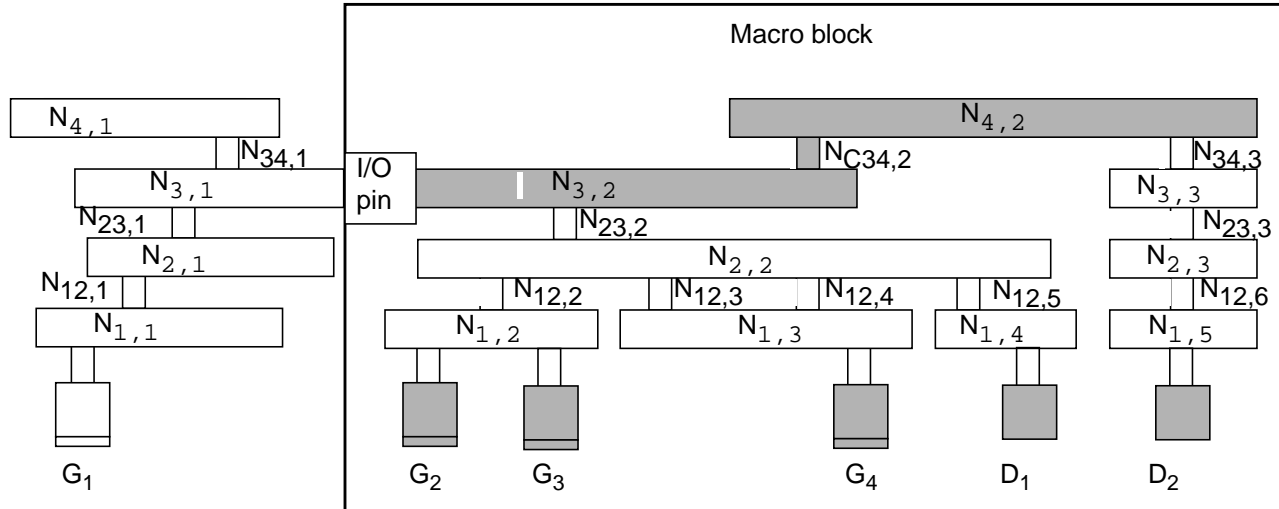
If the keyword ends with ...	It refers to ...	Examples
area sideArea	Drawn area or side area of the metal wires. Measured in square microns.	ANTENNAPARTIALCUTAREA ANTENNAPARTIALMETALAREA ANTENNAPARTIALMETALSIDEAREA ANTENNAPINDIFFAREA ANTENNAPINGATEAREA ANTENNAPINPARTIALCUTAREA
CAR	Relationship the router is calculating CAR is used in keywords for cumulative antenna ratio.	ANTENNAMAXAREACAR ANTENNAMAXCUTCAR ANTENNAMAXSIDEAREACAR ANTENNAPINMAXAREACAR ANTENNAPINMAXCUTCAR ANTENNAPINMAXSIDEAREACAR

Design Example

Figure D-21 on page 221 represents a macro block. This block can be a custom hard block or part of a bottom-up hierarchical flow. The resulting PAE values will be the same in either case. In the example,

- Gates G_1 , G_2 , G_3 , and G_4 are the same size.
- Node $N_{1,3}$ is larger than node $N_{1,2}$.
- Vias (cuts) are all the same size.
- The I/O pin is on *metal3*.
- The area of diffusion for D_1 is `area(Diff1)`.
- The area of diffusion for D_2 is `area(Diff2)`.
- The area of the cut layer that connects node $N_{3,1}$ and node $N_{4,2}$ is `area(NC34,1)`.
- Any damage from the poly layer or poly-to-metal1 via is ignored.

Figure D-21



Relevant Metal Areas

- The relevant metal area for PAE calculations is the partial metal drawn area and side area connected directly to the I/O pin on the inside of the macro on the specified layer.
- Only the same metal layer as the I/O pin or above is needed for PAR calculations in hierarchical designs.

Important

Do not include the drawn area or side area of the I/O pin in the area calculations for the block, because these areas in the calculations for the upper level. Only internal routing area that is not part of the I/O pin should be included.

For the design in the figure above, you must specify values for the following metal areas in the LEF file:

```
ANTENNAPARTIALMETALAREA area(N3,1) LAYER Metal3 ;
ANTENNAPARTIALMETALAREA area(N4,2) LAYER Metal4 ;
ANTENNAPARTIALMETALSIDEAREA sideArea(N3,2) LAYER Metal3 ;
ANTENNAPARTIALMETALSIDEAREA sideArea(N4,2) LAYER Metal4 ;
```

You do not need to specify an ANTENNAPARTIALMETALAREA or ANTENNAPARTIALSIDEMETALAREA for any layer lower than *meta/3* because the I/O pin is on *meta/3*; that is, there is no connection outside the block until *meta/3* is processed.

LEF/DEF 5.4 Language Reference

Calculating and Fixing Process Antenna Violations

Relevant Gate, Diffusion, and Cut Areas

- The relevant gate and diffusion areas are the gate and diffusion areas that connect directly to the I/O pin on the specified layer or are electrically connected to the pin through lower layers.
- The relevant partial cut area is above the current pin layer and inside the macro on the specified layer.

For the example in the figure above, you must specify values for the following gate, diffusion, and cut areas in the LEF file:

```
ANTENNAGATEAREA area(G2 + G3 + G4) LAYER Metal3 ;
ANTENNADIFFAREA area(Diff1) LAYER Metal3 ;
ANTENNADIFFAREA area(Diff1 + Diff2) LAYER Metal4 ;
ANTENNAPARTIALCUTAREA area(N34,2) LAYER via34 ;
```

Calculating the CAR

Use the following keywords to calculate the actual CAR on the I/O pin layer or above.

- The relevant maximum CAR value of the drawn and side areas are from the metal layer that is on or below the I/O pin layer.
- The relevant maximum CAR value of the cut layer is from the cut layer that is immediately above the I/O pin layer.

Use the following keywords to calculate the maximum CAR values:

$$\text{ANTENNAMAXAREACAR} \left(\frac{N_{1,3}}{G_4} + \frac{N_{2,2}}{G_2 + G_3 + G_4} + \frac{N_{3,1}}{G_2 + G_3 + G_4} \right) \text{ LAYER Metal3 ;}$$

$$\text{ANTENNAMAXSIDEAREACAR} \left(\frac{N_{1,3}}{G_4} + \frac{N_{2,2}}{G_2 + G_3 + G_4} + \frac{N_{3,1}}{G_2 + G_3 + G_4} \right) \text{ LAYER Metal3 ;}$$

$$\text{ANTENNAMAXCUTCAR} \left(\frac{N_{12,3} + N_{12,4}}{G_4} + \frac{N_{23,2}}{G_2 + G_3 + G_4} + \frac{N_{34,2}}{G_2 + G_3 + G_4} \right) \text{ LAYER Metal3 ;}$$

Sample LEF File for a Bottom-Up Hierarchical Design

For a macro block like that shown in [Figure D-21](#) on page 221, you should have the following pin information in your LEF file, ignoring `SIDEAREA` values:

PIN example

LEF/DEF 5.4 Language Reference

Calculating and Fixing Process Antenna Violations

```
ANTENNAGATEAREA 0.3 LAYER METAL3 ; # area of G2 + G3 + G4
ANTENNADIFFAREA 1.0 LAYER METAL3 ; # area of D1
ANTENNAPARTIALMETALAREA 10.0 LAYER METAL3 ; # area of N3,1
ANTENNAMAXAREACAR 100.0 LAYER METAL3 ; # max CAR of N3,1
```

```
ANTENNAPARTIALCUTAREA 0.1 LAYER VIA34 ; # area of N34,2
ANTENNAMAXCUTCAR 5.0 LAYER VIA34 ; # max cut CAR of N34,2
```

```
ANTENNAGATEAREA 0.3 LAYER METAL4 ; # area of G2 + G3 + G4
ANTENNADIFFAREA 2.0 LAYER METAL4 ; # area of D1 + D2
ANTENNAPARTIALMETALAREA 12.0 LAYER METAL4 ; # area of N4,2
ANTENNAMAXAREACAR 130.0 LAYER METAL4 ; # max CAR of N4,2
```

END example

Top-Down Hierarchical Design Example

In a top-down design, the router uses the top-level antenna values to check for process antennas inside the block. If the top level is routed first, the top-level routing CAR and PAR values can be passed down into the DEF for the sub-block. This method can also be used to pass down estimated “budgets” for PAR and CAR values.

Following are the keywords to set in the DEF file for the design. In a top-down design you assign a value to the I/O pin that indicates how much routing, CAR, and PAR occurred outside the block already.

```
MACRO macroName
  CLASS BLOCK ;
    PIN pinName
    DIRECTION OUTPUT ;
    [ANTENNAPINPARTIALMETALAREA value [LAYER layerName] ;] ...
    [ANTENNAPINPARTIALMETALSIDEAREA value [LAYER layerName] ;] ...
    [ANTENNAPINGATEAREA value [LAYER layerName] ;] ...
    [ANTENNAPINDIFFAREA value [LAYER layerName] ;] ...
    [ANTENNAPINMAXAREACAR value [LAYER layerName] ;] ...
    [ANTENNAPINMAXSIDEAREACAR value [LAYER layerName] ;] ...
    [ANTENNAPINPARTIALCUTAREA value [LAYER cutlayerName] ;] ...
    [ANTENNAPINMAXCUTCAR value LAYER cutlayerName] ;] ...
  END Z
END macroName
```

LEF/DEF 5.4 Language Reference

Calculating and Fixing Process Antenna Violations

Sample DEF File for a Top-Down Hierarchical Design

An example of the DEF keywords for [Figure D-21](#) on page 221 would be:

```
PINS 100 ;
- example + NET example1
+ ANTENNAPINPARTIALMETALAREA (N3,1) LAYER Metal3 ;
+ ANTENNAPINPARTIALMETALSIDEAREA (N3,1) LAYER Metal3 ;
+ ANTENNAPINGATEAREA (G1) LAYER Metal3 ;
# No ANTENNAPINDIFFAREA for this example

+ ANTENNAPINMAXAREACAR  $\left( \frac{N_{1,1} + N_{2,1} + N_{3,1}}{G_1} \right)$  LAYER Metal3 ;

+ ANTENNAPINMAXSIDEAREACAR  $\left( \frac{N_{1,1} + N_{2,1} + N_{3,1}}{G_1} \right)$  LAYER Metal3 ;

+ ANTENNAPINPARTIALCUTAREA (N34,1) LAYER via34 ;

+ ANTENNAPINMAXCUTCAR  $\left( \frac{N_{34,1} + N_{23,1} + N_{12,1}}{G_1} \right)$  LAYER Metal3 ;

+ ANTENNAPINGATEAREA (G1) LAYER Metal4 ;
+ ANTENNAPINPARTIALMETALAREA (N4,1) LAYER Metal4 ;
+ ANTENNAPINPARTIALMETALSIDEAREA (N4,1) LAYER Metal4 ;
...
END PINS
```


Index

Symbols

,... in syntax [11](#)
 ... in syntax [10](#)
 [] in syntax [10](#)
 {} in syntax [10](#)
 | in syntax [10](#)

A

abutment pins [53](#)
 alias
 expansion in DEF and LEF [87](#)
 names in DEF and LEF [86](#)
 statements in LEF and DEF [86](#)

B

bitmapped via pattern convention [165](#)
 blockages, simplified [48](#)
 Boolean expressions, in DEF and LEF [81](#)
 braces in syntax [10](#)
 brackets in syntax [10](#)
 BUSBITCHARS statement
 description, DEF [99](#)

C

capacitance
 peripheral [29](#), [57](#)
 wire-to-ground [28](#), [57](#)
 case sensitivity, in LEF file [56](#)
 cell modeling
 combining blockages [48](#)
 symmetry in LEF [43](#)
 characters
 escape [12](#)
 information [11](#)
 COMPONENTS statement
 description, DEF [100](#)
 compressed via pattern convention [166](#)
 conventions
 user-defined arguments [10](#)

user-entered text [10](#)
 COVER model, definition in LEF [39](#)

D

database
 converting LEF values to integer
 values [65](#)
 debugging
 DEF files [90](#)
 LEF libraries [90](#)
 parametric macros [90](#)
 DEF
 DEFINE statements [79](#)
 example [157](#)
 line length [97](#)
 sections [97](#)
 syntax overview [97](#)
 DEF syntax
 PINS [117](#)
 DEF syntax and description
 BUSBITCHARS [99](#)
 COMPONENTS [100](#)
 DESIGN [102](#)
 DIEAREA [103](#)
 DIVIDERCHAR [16](#), [103](#)
 GCELLGRID [105](#)
 GROUPS [106](#)
 HISTORY [108](#)
 NETS [108](#)
 PINPROPERTIES [124](#)
 PROPERTYDEFINITIONS [125](#)
 REGIONS [128](#)
 ROW [129](#)
 SPECIALNETS [135](#)
 TECHNOLOGY [142](#)
 TRACKS [143](#)
 UNITS DISTANCE MICRONS [143](#)
 VERSION [144](#)
 VIAS [145](#)
 DEFINE statements [79](#)
 Boolean expressions [81](#)
 examples [85](#)
 expansion characters [84](#)
 LEF and DEF variable names [79](#)

LEF/DEF 5.4 Language Reference

- numeric expressions [80](#)
- restrictions [84](#)
- string expressions [81](#)
- substituting for tokens [84](#)
- syntax [83](#)

DEFINE syntax, in DEF and LEF [83](#)

DESIGN statement
description, DEF [102](#)

diagonal vias, recommendation for
RGrid [34](#)

DIEAREA statement
description, DEF [103](#)

DIVIDERCHAR statement
description, DEF [16](#), [103](#)

E

edge capacitance [29](#)

EEQ statement, LEF syntax [40](#)

electrically equivalent models, LEF
syntax [40](#)

endcap models, definition in LEF [40](#)

equivalent models
electrically equivalent (EEQ) [40](#)
logically equivalent (LEQ) [41](#)

error checking, utilities [90](#)

escape character [12](#)

expressions, in DEFINE statements [80](#)

F

feedthrough pins
LEF [54](#)

forbidden via locations [69](#)

FOREIGN references
LEF syntax [40](#)
offset between LEF and GDSII [40](#)

G

GCell grid
INITIALIZE FLOORPLAN
command [106](#)
restrictions [106](#)
uniform, in DEF [105](#)

GCELLGRID statement
description, DEF [105](#)

GROUPS statement

description, DEF [106](#)

H

HISTORY statement
description, DEF [108](#)

I

INOUT pins, netlist [49](#)

INPUT DEF command
error checking [90](#)

INPUT GDSII command
with incremental LEF [89](#)

INPUT LEF command
error checking [90](#)
incremental capability [89](#)

INPUT pins, netlist [49](#)

italics in syntax [10](#)

L

LAYER (nonrouting) statement
description, LEF [17](#), [21](#)

layers
for LEF via descriptions [68](#)
routing order in LEF [18](#), [22](#), [24](#)

LEF
DEFINE statements [79](#)
example [147](#)
files
distance precision [14](#)
line length [14](#)
overview [14](#)
routing layer order [18](#), [22](#), [24](#)

LEF syntax
overview [14](#)

LEF syntax and description
LAYER, nonrouting [17](#), [21](#)
MACRO [38](#)
NAMESCASESENSITIVE [56](#)
NONDEFAULT rule [56](#)
OBS, macro obstruction [46](#)
PIN macro [49](#)
PROPERTYDEFINITIONS [60](#)
SITE [63](#)
SPACING [62](#)
UNITS [64](#)

LEF/DEF 5.4 Language Reference

VERSION [67](#)

VIA [67](#)

VIARULE [71](#)

VIARULE viaRuleName

GENERATE [74](#)

LEF values converted to integer values [65](#)

legal characters [11](#)

LEQ statement, logically equivalent models
in LEF [41](#)

library design, simplifying blockages [48](#)

literal characters [10](#)

logically equivalent models, LEQ syntax in
LEF [41](#)

M

macro obstruction, OBS statement
description, LEF [44](#), [46](#)

macro PIN statement
description, LEF [49](#)

MACRO statement
description, LEF [38](#)

MAXHALPERIMETER, DEF groups
parameter [107](#)

MAXX, DEF groups parameter [107](#)

MAXY, DEF groups parameter [107](#)

models, site orientation [42](#)

mustjoin pins [52](#)

N

names, variable in DEFINE statements [79](#)

NAMESCASESENSITIVE statement
description, DEF [108](#)

netlist pins
INOUT [49](#)
INPUT [49](#)
OUTPUT [49](#)

nets
changing regular nets to special [141](#)
mustJoin nets [110](#)
same-net spacing, LEF [62](#)

NETS statement
description, DEF [108](#)

NONDEFAULT rule statement
description, LEF [56](#)
spacing [58](#)

O

OBS (macro obstruction) statement
description, LEF [44](#), [46](#)

obstructions, simplified [48](#)

offset between LEF and GDSII [68](#)

Or-bars in syntax [10](#)

orientation
models [42](#)
pin [102](#), [116](#), [122](#)

OUTPUT pins, netlist [49](#)

overlaps, specifying in LEF [49](#)

P

peripheral capacitance [29](#)

PIN (macro) statement
description, LEF [49](#)

PINPROPERTIES statement
description, DEF [124](#)

pins
abutment [53](#)
direction in LEF [51](#)
external, DEF [118](#)
feedthrough pins in LEF [54](#)
INOUT [49](#)
INPUT [49](#)
modeling in LEF [49](#)
mustjoin [52](#)
netlist [49](#)
orientation [102](#), [116](#), [122](#)
OUTPUT [49](#)
power geometries [54](#)
ring [54](#)
using in LEF [55](#)

PINS statement
syntax, DEF [117](#)

PITCH parameter, ratio in three-layer
design [34](#)

placement site function, SITE statement in
LEF [63](#)

ports
allowing shorts [53](#)
geometries in LEF [53](#)
layers in LEF [53](#)
in LEF [53](#)
multiple pins [53](#)
width in LEF [53](#)

power pin

LEF/DEF 5.4 Language Reference

geometries in LEF [54](#)
PROPERTYDEFINITIONS statement
description, DEF [125](#)
description, LEF [60](#)

R

REGIONS statement
description, DEF [128](#)
regular expressions [12](#)
regular wiring
changing to special [141](#)
orthogonal paths [116](#), [140](#)
RGrid, description [34](#)
RING models, definition in LEF [39](#)
ring pins [54](#)
routing time, diagonal vias [34](#)
routing width, LEF syntax [33](#)
ROW statement
description, DEF [129](#)

S

same-net spacing [62](#)
scan chains
example [162](#)
rules [132](#)
SI units in LEF [65](#)
SITE statement
description, LEF [63](#)
sites
symmetry [64](#)
spacing
same-net [62](#)
SPACING statement
description, LEF [62](#)
same-net [62](#)
special wiring
changing from regular to special [141](#)
description [138](#)
pins and wiring, DEF [140](#)
via rules [71](#)
width in database [138](#)
SPECIALNETS statement
description, DEF [135](#)
stacked vias, same-net spacing rule [62](#)
symmetry in LEF, cell modeling [43](#)
syntax conventions [10](#)

T

TECHNOLOGY statement
description, DEF [142](#)
three-layer design, pitch ratio [34](#)
TRACKS statement
description, DEF [143](#)
turn vias, rules [77](#)

U

UNITS DISTANCE MICRONS statement
description, DEF [143](#)
UNITS statement
description, LEF [64](#)

V

values in library database [65](#)
variable names, DEFINE statement [79](#)
VERSION statement
description, DEF [144](#)
vertical bars in syntax [10](#)
via pattern convention
bitmapped [165](#)
bitmapped example [167](#)
character sets [167](#)
choosing [165](#)
compressed [166](#)
compressed example [168](#)
environmental variable [165](#)
overview [165](#)
VIA statement
description, LEF [67](#)
VIARULE statement
description, LEF [71](#)
VIARULE viaRuleName GENERATE
statement
description, LEF [74](#)
vias [68](#)
default via rules [73](#)
default vias in LEF [67](#)
layers for vias in LEF [68](#)
locations, forbidden [69](#)
same-net spacing [62](#)
special wiring [71](#)
stacked [62](#)
turn via rules [77](#)

VIAS statement
description, DEF [145](#)

W

wide wire signal wire, specifying [56](#)
wiring, regular
 changing to special [141](#)
 orthogonal paths [116](#), [140](#)
wiring, special
 changing from regular to special [141](#)
 description [138](#)
 pins and wiring [140](#)
 via rules [71](#)
 width in database [138](#)