cadence®

Dracula User Guide

Product Version IC23.1
September 2023

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

Contents

Preface 9
Related Documents 9
Typographic and Syntax Conventions 10

1
Interfaces to Dracula 11
Overview of This Chapter 11
Using Cadence Data in Dracula 11
Preparing To Use a DFII Database 12
Specifying the Path to Your Layout 12
Controlling Case Sensitivity 14
Extracting Objects from a DFII Layout 15
Sample Rules File 16
Converting a GDSII Rules File for Cadence Input 17

2
Writing Rules for Dracula 19
Writing Dracula Rules Files 19
Using Rules File Examples 20
Techniques for Preparing Your Rules File 21
Writing DRC Checks 22
Writing Antenna Checks 22
Writing In-the-Direction-Of Checks 31
Writing Exact Contact Size Checks 35
Writing Metal Reflection/ Crosstalk Checks 36
Writing Electromigration Checks 36
Writing Corner Checks 45
Writing Edge Checks 45
Defining Devices 48
Connecting Your Network 50

Determining the Master Layer 50
Connecting Multiple Layers 51
Preparing To Use Dracula Interactive 52

3
Working With 32 and 64-bit Applications 53

4
Selecting a Run Mode 57
About Run Modes 57
Using Flat Mode 58
Running Dracula in Flat Mode 58
When To Use Flat Mode 58
Verifying Gate Arrays 60
Texting for Flat Mode 60
Using Dracula Distributed Processing 60
When To Use Dracula Distributed Processing 60
Using Hierarchical Mode 61
Running Dracula in Hierarchical Mode 61
What the Hierarchical Design Rule Checker Checks 61
When To Use the Hierarchical Design Rule Checker 62
Using Multilevel Mode 62
Running Dracula in Multilevel Mode 63
Selecting Hcells for a Multilevel Run 63
When To Use Multilevel Mode 63
Using Cell Mode 63
Running Dracula in Cell Mode 64
When To Use Cell Mode 64
Texting for Cell Mode 65
Using Composite Mode 70
Running Dracula in Composite Mode 70
Checks in Composite Mode 71
When To Use Composite Mode 71
Composite Mode and Blackbox LVS 72
Texting for Composite Mode 72

Improving Performance 74
Grouping Functions 74
Preallocating Swap Space 75
Optimizing Rules 76
X-to-Y Ratio of the Circuit 76

5
Extracting RC Parasitics 77

About RC Extraction 77
Ensuring Accuracy in RC Extraction 78
Providing Constants 78
Adjusting Your Design Database for Fabrication Effects 78
Using Equations with Flexible LPE 79
Extracting Parasitic Capacitance 80
Preparing To Extract Capacitance 81
Constants for Debugging Capacitance 84
Extracting Overlap and Sidewall Capacitance 85
Extracting Directional Sidewall Capacitance 87
Correcting for Colinear Sidewall Capacitance 90
Extracting Single-Layer Fringe Capacitance 94
Extracting Two-Layer Fringe Capacitance 95
Extracting Over-the-Cell Routing Wire Overlap Capacitance with Cell Geometries in HLPE 96
Extracting Parasitic Resistance 98
Preparing to Extract Resistance 99
Creating Resistor Recognition Layers 103
Extracting Simple Metal and Poly Resistors 104
Extracting Two-Layer Metal Parasitics 105
Extracting Contact Resistors 108
Extracting Diffusion Resistors 111
Correcting Sidewall Capacitance 112
Extracting Fringe Capacitance in PRE 114
Controlling How Contacts Are Cut 115
Extracting 2.5D MB Parasitics 117
Dracula 2.5D MB Capabilities 118

Using Piecewise Analysis . . . 118
Extracting Sidewall Capacitance with Fringing Effects . . . 118
Using One Function to Extract RC Parasitics . . . 120
Defining Parasitics Layers . . . 120
Extracting Overlap and Sidewall Capacitance . . . 121
Extracting Directional Sidewall and Colinear Edge Capacitance . . . 121
Extracting 2D3B Capacitance . . . 121
Extracting One- and Two-Layer Fringe Capacitance . . . 121
Modifying the Coefficient Generator Interface with Dracula to Improve LPE Accuracy . . . 122
Extracting Sheet Resistance . . . 126
Extracting Multiple Resistance on the Same Metal Layer . . . 126
Extracting Contact Resistance . . . 127
Capacitance or Resistor Model in the Extracted Netlist . . . 127
Function Syntax . . . 128
Before You Start . . . 130
Creating Additional Subtypes . . . 130
Examples . . . 131
Extracting a Single Net . . . 133
Before You Start . . . 133
Coding Single-Net Extraction . . . 134
The Dracula To RCX (dracToRcx) Interface . . . 135
Module Descriptions . . . 136
RCX Commands . . . 140
Running the DracToRCX Interface . . . 141

6
Setting up Hierarchical Dracula . . . 145
About Hierarchical Dracula . . . 145
Prerequisites . . . 146
Hierarchical Products and Modes . . . 146
Running a Hierarchical Design Rule Check . . . 146
Writing Hierarchical Design Rules . . . 147
Checking Memories and Arrays . . . 153
Using HDRC Output . . . 154

Selecting Hcells 158
Selecting Hcells Automatically 158
Selecting Hcells Manually To Improve Performance 160
Running a Multilevel Design Rule Check 161
Understanding Multilevel Processing 162
Running a Multilevel Hierarchical DRC 163
Running a Hierarchical Electrical Rules Check 163
Writing Hierarchical Electrical Rules 163
Deciding What To Check 165
Connecting the Network 165
Flattening Power and Ground Text 166
Excluding Hcell Geometries in HERC 166
Using Composite Plane Geometries in Hcells 167
Using HERC Output 168
Running a Hierarchical Network Comparison 171
Writing Hierarchical Network Comparison Rules 171
HLVS Prerequisites 173
Excluding Hcell Geometries in HLVS 174
Using Cell Boundaries 175
Marking Feedthroughs on the Layout 176
Using a Cross-Reference File for Hcell Names 177
Preparing Your Netlist 177
Compiling Your Netlist 179
Running Hierarchical Parameter Extractions 179
About Hierarchical Parameter Extraction 179
Writing Parameter Extraction Rules 181
Using a Cross-Reference File for Hcell Names 182
Using HLPE To Extract a Complete Netlist 182
Making Hierarchical Dracula Run Faster 183
Maximizing Operations 183
Minimizing Redundant Checking 183
Altering the Hcell Selection Criteria 184

Index 187

Preface

This manual assumes that you are familiar with the development and design of integrated circuits. It contains task-related information about the Dracula® standalone verification tool.

Dracula is an integrated system that offers a full tool suite to meet most verification needs.

- Design Rules Check (DRC) to ensure the chip is processed correctly
- Layout Versus Schematic (LVS) and Electrical Rules Check (ERC) to ensure the chip is electrically correct
- Layout Parameter Extraction (LPE) and Parasitic Resistance Extraction (PRE) parasitic data that is output to simulators to ensure the chip performs to specifications
- DracToRCX interface to allow Dracula users the ability to run parasitic extraction through the Assura RCX tool
- Mask generation for accuracy

These tools work from the same database to ensure a smooth transition through each phase of IC verification.

This preface discusses the following:

- Related Documents
- Typographic and Syntax Conventions

Related Documents

For more information about Dracula and other related products, you can consult the sources listed below.

- The *Cadence Installation Guide* tells you how to install the product.
- The *Dracula Standalone Verification Reference Manual* gives complete reference information on the Dracula suite of tools and functions.

Typographic and Syntax Conventions

This list describes the syntax conventions used in this book.

`LITERAL`	UPPERCASE words indicate keywords that you must enter literally. These keywords represent function or option names.
`argument`	Words in italics indicate user-defined arguments for which you must substitute a name or a value.
`...`	Three dots (...) indicate that you can repeat the previous argument.
`; <functions omitted>`	When you see this notation in a rules file listing, it means that lines have been omitted from the rules file. The complete rules files are not included in examples. Only the functions needed to demonstrate a specific rule are listed.

1

Interfaces to Dracula

The Dracula® standalone verification product offers a suite of software applications for Integrated Circuit (IC) design verification.

Overview of This Chapter

This chapter contains the following sections.

- Using Cadence Data in Dracula
  - Preparing To Use a DFII Database
  - Specifying the Path to Your Layout
  - Controlling Case Sensitivity
  - Extracting Objects from a DFII Layout
  - Sample Rules File
  - Converting a GDSII Rules File for Cadence Input

Using Cadence Data in Dracula

You can create a layout with a Cadence® design framework II (DFII) tool, such as Virtuoso® layout editor and use Dracula to verify the layout database.

When you put the `SYSTEM=CADENCE` function in the Description block of your Dracula rules file, the CDSIN module processes your DFII database and converts it to Dracula internal format.

Although you can use the Virtuoso tools to generate the layout database in a standard format, such as GDSII, it is quicker to generate it in Cadence DFII format.

In this section, you'll learn how to

- Use a Cadence-format layout database as input to Dracula
- Convert an existing Dracula rules file that uses GDSII input to use Cadence-format input

Preparing To Use a DFII Database

To use CDSIN to convert a DFII database to Dracula internal format, you must meet the following requirements:

- You must have the DFII license server daemon, `cdsd`, running on the following machines:
  - The machine where you start your Dracula job
  - The machine where any libraries you reference are located

    You reference the libraries with the `INDISK-FILE` function in your Dracula rules file:

    ```
    INDISK-FILE = indisk.file
    ```

    For more information about `INDISK-FILE`, see "Specifying the Path to Your Layout" on page 12. You can soft-mount or hard-mount the libraries to the machine where you run the job.
- For runs with parameterized cells (Pcells) or reference libraries, you also need the following two files.
  - `dbRead.cxt` ensures that cells and reference libraries are read in correctly. This is the only file needed in the `/context` directory for CDSIN.

    If you have installation problems, copy `dbRead.cxt` from the DFII installation hierarchy (*`DFIIhier`*`/bin/../etc/context/`) to the Dracula installation hierarchy (*`dracdir`*`/bin/../etc/context/`). The hierarchical relationship between Dracula `bin` and `etc` directories must be the same as the DFII `bin` and `etc` directories. CDSIN determines the DFII installation path internally.
  - `instdir` is an installation marker file that CDSIN needs. It must be in the same directory (*`dracdir`*) as CDSIN.

Specifying the Path to Your Layout

You can specify the path to your layout database using the `INDISK` and `INDISK-FILE` functions in the Description block of your Dracula rules file.

- `INDISK`

If you have only one library path, use `INDISK = directoryname`.

- `INDISK-FILE`

  If your rules file references more than one library, use `INDISK-FILE = filename`. For example, the contents of *`filename`* might be

  ```
  ./myLib1
  /mnt/Lib2
  /mnt/Lib3
  ```

  You can reference libraries mounted across the network. For the previous example, a UNIX `df` command might print the following:

  ```
  wkstnA:/usr2/lib2 949180  707444  222753 76% /mnt/Lib2
  wkstnB:/usr3/lib3 961501  785590  156681 83% /mnt/Lib3
  ```

To specify the name of the DFII library containing the primary cell in your design, use the `LIBRARY` function in the Description block of your rules file.

```
LIBRARY = mpuLib
```

The `LIBRARY` function is required when you use the `SYSTEM=CADENCE` function.

- `IMULTI-GDS2-IN`

In the 4.7 and previous versions, the GDSII files were limited to 2.0 Gigabytes.

An enhancement in the 4.8 release suppresses this limitation. To work around the 2GB limit in streamout, Dracula now reads GDSII files that have been split into multiple files on a disk. The files can be split by cells or layers.

Example

```
MULTI-GDS2-IN = inFile CELL/LAYER
```

The contents of `inFile` might be

```
./CellA.gds
./CellB.gds
./CellC.gds
```

or

```
./Layer1.gds
./Layer2.gds
./Layer3.gds
```

The `CELL/LAYER` option is indicates that the `.gds` files are split by cell or by layer. You must specify either `CELL` or `LAYER`.

Controlling Case Sensitivity

To ensure correct results, text in your layout must match text in your schematic. Cell names, labels, and pins in your DFII layout can be upper-, lower-, or mixed-case. By default, Dracula treats all the cell, pin, and label names as case sensitive. When you use `SYSTEM = CADENCE` in your Dracula rules file, the `CNAMES-CSEN = YES` option (the default) applies to

- All signal and cell names, not just those specified in the Description block
- All modules

To convert everything to uppercase, specify `CNAMES-CSEN = NO`. To control the case of text names separately, use the `TNAMES-CSEN` function. For an explanation of how to use these two functions together, see "CNAMES-CSEN" in the "Description Block Commands" chapter of the *Dracula Reference.*

If you are using external files for Hcells, corresponding points, and composite and cell texts for LVS, LPE, and PRE, make sure the text is consistent with the `CNAMES-CSEN` command.

Controlling Case in Schematic Netlists

When you compile your schematic netlist with LOGLVS, activate case-sensitive mode by using the `CASE` function before LOGLVS reads in your circuit. By default, LOGLVS converts all signals to uppercase.

When CDLOUT creates a schematic netlist, it keeps the case information. CDLOUT netlists global signals such as `gnd!` and `vdd!` in the `*.GLOBAL` line.

Correcting Case Problems

If you were not aware or not in control of case requirements when the schematics and layout were created, you can correct problems with cell and signal names as follows:

- Correcting cell names

  If the case of Hcell names is different in the schematic and layout, use an Hcell file (`HCELL-FILE = fileName`) to match the Hcell names in both places.

- Correcting signal names

  If the case of the signal names is different in the schematic and layout, use the `EDTEXT` and `HEDTEXT` files to specify layout text that matches text in the schematic. When you specify text in a text file with the same set of coordinates as text on the layout, the correct file text overwrites the incorrect layout text.

If necessary, you can edit the CDL netlist to convert the cell names to the same case.

Extracting Objects from a DFII Layout

To extract objects from your DFII layout, specify layer purpose pairs in the Input-Layer block of your Dracula rules file. If you don't specify a purpose for a DFII layer, the default is that all purposes except `pin` on that layer are read into the Dracula database.

```
; all the layer purposes on the active layer will be
; read into the Dracula ACTIVE layer
ACTIVE = active
```

If you specify a purpose, then only that purpose is read in.

```
; only the maskAdd layer purpose of the poly layer
; is read into the Dracula POLY layer
POLY   = poly maskAdd
```

For more information about specifying DFII input layers, see "Layer Name Definition" in the "Input-Layer Block Commands" chapter of the *Dracula Reference.*

Pins in a DFII layout database mark connection points to interconnect layers. To read in the pin information as text, specify the pin layer with the `TEXT` function in the Input-Layer block of your rules file.

The layer function syntax for DFII layers is

layer_name = *dfII_layer* [*purpose*] TEXT=*layer2* [ATTACH=*layer3*]

layer_name
: A name of your choice.

***dfII_layer* [*purpose*]**
: If you use the `TEXT` keyword, this is the DFII layer purpose pair that contains the pins. The layer purpose pair must be defined in the DFII library technology file.

TEXT
: A keyword that tells Dracula to generate text from pins.

layer2
: A layer name/number of your choice. It does not have to be a layer used in the Dracula rules file or defined in the technology file.

ATTACH=*layer3*
: The layer on which Dracula attaches the text.

Dracula reads in pin purpose shapes only when you specify a pin purpose to create text from pins. When the pins are on a design layer purpose, you must define a separate layer for the text, as the following example shows.

```
mettxt1 = metal1  pin  TEXT = dummy  ATTACH = metal1
```

The `TEXT` function puts all *metal1* pins on `pin` purposes on the *mettxt1* layer. Text is generated and attached to *metal1* and paired with

```
metal1 = metal1
```

which is necessary to read in all the drawing information. Similarly, if you specify

```
metal1 = metal1 drawing TEXT = dummy ATTACH = metal1
```

all *metal1* pins on the drawing purpose are read in. Text is generated and attached to *metal1*.

Each text layer corresponds to one pin layer.

Sample Rules File

The following sample rules file demonstrates functions you might need to read in a DFII layout database.

```
*DESCRIPTION
;
SYSTEM          = CADENCE
LIBRARY         = prelib
OUTDISK         = lpe1
PRIMARY         = lpe1 layout
SCALE           = 0.001  MICRON
RESOLUTION      = 0.001  MICRON
; INDISK-FILE is used when more than one library is
; referenced. If there is just one library path, use the
; INDISK = directoryname function
INDISK-FILE = ../indisk.file
;
; <functions omitted>
;
*END
*INPUT-LAYER
;
; In the following, active, poly, pimplant, metal, pin,
; and so on are layer purpose pairs defined in the DFII
; technology file.
;
ACTIVE        = active
POLY          = poly maskAdd; read in only the maskAdd purpose
PIMP          = pimplant
PWELL         = pwell
METAL1        = metal1
METAL2        = metal2
;
; <functions omitted>
;
```

```
; Read in all the pins and treat as text attached to METAL1
; and METAL2 -- mettxt1 for METAL1, mettxt2 for METAL2,
; where the pins are drawn on the pin purpose layer
;
mettxt1 = metal1 pin text=dummy attach=METAL1
mettxt2 = metal2 pin text=dummy attach=METAL2
;
SUBSTRATE = BULK 63
;
CONNECT-LAYER = NSUB PWELL NSD PSD POLY METAL
;
*END
*OPERATION
;
; <functions omitted>
;
*END
```

Converting a GDSII Rules File for Cadence Input

You might already have a Dracula rules file for your process technology that specifies a GDSII input database. To convert this rules file for Cadence input, change the Description block of your rules file as follows.

- Change `SYSTEM` from `GDS2` to `CADENCE`
- Change `INDISK` from the layout database file name to the name of your DFII library directory, or replace `INDISK` with `INDISK-FILE` and specify the name of a file that lists all of your reference libraries. See "Specifying the Path to Your Layout" on page 12.
- Add the `LIBRARY` function to specify the name of your DFII library.

In the Input-Layer block, change the GDSII layer numbers to the names of layers in your technology file. You can also include some or all of your layer purposes in the Dracula verification, as described in "Extracting Objects from a DFII Layout" on page 15. The default layer purpose is `pin`.

GDSII-to-Cadence Conversion: Description Block

GDSII	Cadence
`PRIMARY = MPU4f`	`PRIMARY = MPU4f`
`INDISK = mpu4f.gds2`	`INDISK-FILE = indisk.fil`
	`LIBRARY = mpuLib`
`OUTDISK = DRCOUT`	`OUTDISK = DRCOUT`
`MODE = EXEC NOW`	`MODE = EXEC NOW`
`SYSTEM = gds2`	`SYSTEM = cadence`

GDSII-to-Cadence Conversion: Description Block, *continued*

GDSII	Cadence
SCALE = .001 MICRON	SCALE = .001 MICRON
RESOLUTION = .001 MICRON	RESOLUTION = .001 MICRON
PROGRAM-DIR = /net/cds000/hp2/dracula	PROGRAM-DIR = /net/cds000/hp2/dracula
PRINTFILE = mpu4f4	PRINTFILE = mpu4f4

GDSII-to-Cadence Conversion: Input-Layer Block

GDSII	Cadence
PWELL = 1	PWELL = pwell
ACTIVE = 2	ACTIVE = diff
PDIFS = 3	PDIFS = pdiff
NDIFS = 4	NDIFS = ndiff
POLY = 5	POLY = poly1
PIMPL = 6	PIMPL = pimplant
CONTS = 7	CONTS = cont
METAL1 = 8 TEXT = 60	METAL1 = metal1 TEXT = intext
VIA = 9	VIA = via
METAL2 = 10	METAL2 = metal2
SUBSTRATE = BULK 63	SUBSTRATE = BULK 63

2

Writing Rules for Dracula

In this chapter, you'll learn about the following:

- Writing Dracula Rules Files
- Writing DRC Checks
  - Writing Antenna Checks
  - Writing In-the-Direction-Of Checks
  - Writing Exact Contact Size Checks
  - Writing Metal Reflection/ Crosstalk Checks
  - Writing Electromigration Checks
  - Writing Corner Checks
  - Writing Edge Checks
- Defining Devices
- Connecting Your Network
- Preparing To Use Dracula Interactive

Writing Dracula Rules Files

You write a Dracula rules file to verify designs in your specific process technology. Although your technology might be unique, it will have some features in common with similar technologies. The *Dracula Reference* contains many good examples of complete rules files for a variety of technologies and purposes. You can use these rules as guidelines when you write your own rules files.

Using Rules File Examples

For general design rule techniques, see "Creating a Rules File" in the "Using Dracula" chapter of the *Dracula Reference.* This section gives a basic CMOS example. In addition, you can refer to other rules file examples listed in the following table.

Note: All examples listed in the table can be found in the *Dracula Reference.*

Sample Dracula Rules Files

Rules	Technology	Input format	Contents	Reference manual section
DRC	NMOS	Applicon	Design rules for gate, depletion, external separation, enclosure, and metal reflection checks	"DRC Example" in the "Checking Design Rules (DRC)" chapter
ERC	NMOS	GDSII	Extracts depletion and enhancement transistors	"NMOS ERC Rules File" in the "Checking Electrical Rules (ERC)" chapter
ERC	CMOS	GDSII	Extracts N-channel and P-channel transistors	"CMOS ERC Rules File" in the "Checking Electrical Rules (ERC)" chapter
LVS	Bipolar	GDSII	Extracts vertical NPN and PNP transistors, lateral PNP transistors, Schottky diodes, metal-to-N+ capacitors, and N+ and P resistors	"Bipolar LVS Example" in the "Comparing Layouts and Schematics (LVS)" chapter
LPE	NMOS	GDSII	Extracts depletion and enhancement transistors, a parasitic diode, and parasitic capacitors	"NMOS LPE Input File" in the "Extracting Electrical Parameters (LPE)" chapter
LPE	CMOS	GDSII	Extracts N-channel and P-channel transistors and parasitic diodes and capacitors	"CMOS LPE Input File" in the "Extracting Electrical Parameters (LPE)" chapter

Sample Dracula Rules Files, *continued*

Rules	Technology	Input format	Contents	Reference manual section
PRE	CMOS	GDSII	Extracts N-channel and P-channel transistors and parasitic diodes, resistors, and capacitors	"Sample PRE Rules File" in the "Extracting Parasitic Resistance (PRE)" chapter

You might also want to refer to the following reference manual sectios on database conversion.

- Database conversion

  The "CONVERT-DATABASE" section in the "Description Block Commands" chapter contains rules files for the following database format conversion functions:

  - Example: Applicon to GDSII
  - Example: GDSII to Applicon

Techniques for Preparing Your Rules File

In this chapter, you will learn the following techniques for writing your rules file:

- Writing DRC checks

  Some design rule checks are similar in all process technologies. You can use "Writing DRC Checks" on page 22 to learn how to write antenna checks, electromigration checks, and other types of checks you might need to make.

- Defining devices

  The rules that you write to extract devices in your technology will probably be similar to the rules in the *Dracula Reference*. Each process technology can be unique, however, the Defining Devices section gives general information to guide you in extracting devices in your technology.

- Connecting your network

  When you write rules to create a network in your design, you use the `CONNECT` and `CONNECT-LAYER` functions. You need to know how Dracula uses these functions to connect layers. For more information, see Connecting Your Network section.

- Preparing to use Dracula Interactive

  If you are planning to use Dracula Interactive to view the results of your Dracula job, there are several prerequisites. See the "Preparing To Use Dracula Interactive" section for a list of requirements.

Writing DRC Checks

In this section, you'll learn how to write the following types of checks:

- Antenna checks
- In-the-direction-of checks
- Exact size checks
- Electromigration checks
- Corner checks
- Edge checks

For examples of common checks, see the "DRC Example" section in the "Checking Design Rules (DRC)" chapter of the *Dracula Reference*.

Writing Antenna Checks

You use antenna checks to check each node for the area ratio of geometries on a given layer to devices on the same node. The area ratio of devices affects the drive current and transition time. An antenna check ensures that induced capacitance is under a certain value so that electrical characteristics stay stable.

The most common antenna check compares the ratio of field poly area to gate area on each input node. You can write similar checks that compare metal, diffusion, and contact layer area to gate area.

You can write an antenna check rules file without using the LPECHK, LVSCHK, or LPESELECT functions, which shortens the run time. The functions you might need for an antenna check are listed in the following table.

Functions for Antenna Checks

Function	Description	Rules file block
PARSET	Lists the set of parameters that you want to extract, such as area and perimeter.	*DESCRIPTION
STAMP, AND, or NOT	Passes node information needed by LEXTRACT from one layer to another.	*OPERATION
LEXTRACT	Creates a layer that defines the parameters specified by PARSET.	
COMPUTE	Calculates the value and ratio of the geometric parameters you extract with LEXTRACT. You can use this function instead of CALCULATE. See "Extracting Area Ratios Using the COMPUTE Function."	
CALCULATE	Calculates the value and ratio of the geometric parameters you extract with LEXTRACT. You can use this function instead of COMPUTE. See "Extracting Area Ratios Using the CALCULATE Function."	
CHKPAR	Uses values calculated by a previous COMPUTE or CALCULATE function to report data based on your criteria.	

Extracting Area Ratios Using the COMPUTE Function

You use the COMPUTE function to compute values based on a criterion that you specify. The criterion can be MIN, MAX, SUM, or MINGATE. The CHKPAR function uses the values that COMPUTE extracts to report the data you select.

You use logical operations and LEXTRACT to prepare layers to use in COMPUTE, as shown in the following example.

```
;
; <functions omitted>
;
PARSET ANT AREA PERI
;
; <functions omitted>
;
NOT   POLY  GATE  FPOLY ; field poly
STAMP GATE  BY    POLY
```

```
STAMP FPOLY BY    POLY
LEXTRACT ANT GATE  BY NODE XGATE
LEXTRACT ANT FPOLY BY NODE XFPOLY
```

Use the *XGATE* and *XFPOLY* layers in `COMPUTE` operations, as shown in the sections that follow.

Using the MIN Option

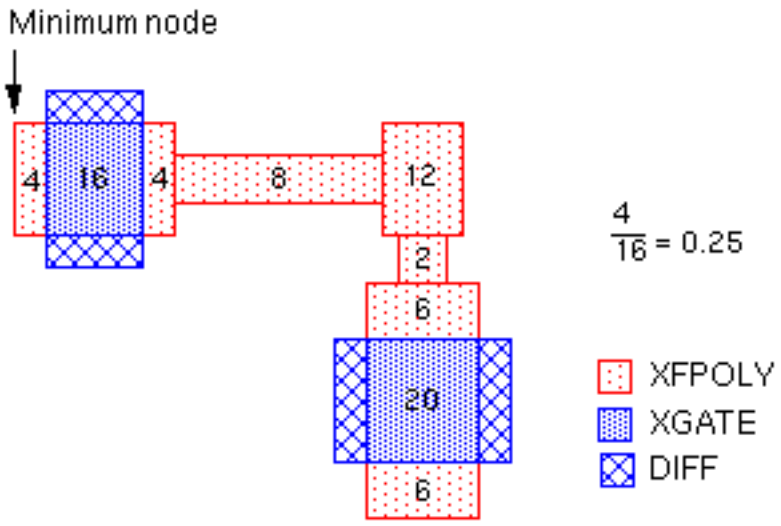
For the following check

```
COMPUTE MIN XFPOLY XGATE ANTENNA
```

the `MIN` option uses the following equation:

$$\frac{\text{minimum area of XFPOLY on the node}}{\text{minimum area of XGATE on the node}} = \text{ratio}$$

The minimum node in the following figure has a ratio of 0.25. For this node, the minimum area for a separate region of *XFPOLY* is 4. The other regions are 6 and 32 (4 + 8 + 12 + 2 + 6).



If you use only one input parameter file, the `COMPUTE MIN` function calculates the value for each separate area. For example, the following `COMPUTE` function reports the minimum separate area for each node on the *XFPOLY* layer.

```
; <functions omitted>
PARSET ANT AREA
; <functions omitted>
```

```
LEXTRACT ANT FPOLY BY NODE XFPOLY
COMPUTE  MIN XFPOLY ANTENNA
```

Using the MAX Option

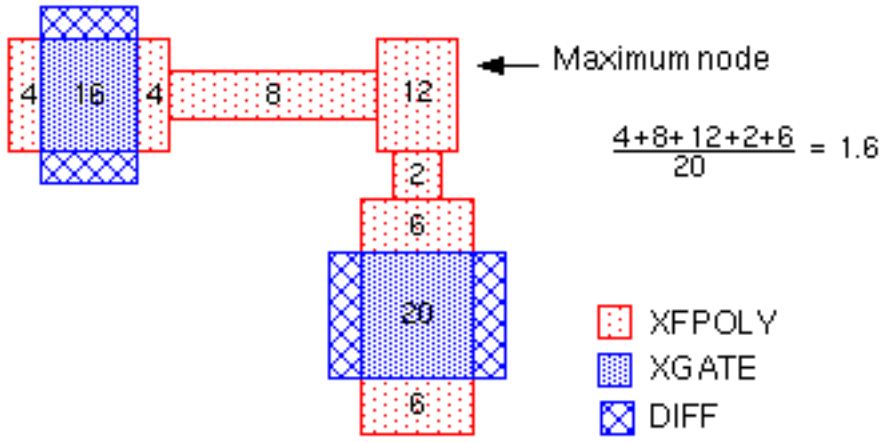
For the following check

```
COMPUTE MAX XFPOLY XGATE ANTENNA
```

the `MAX` option uses the following equation:

$$\frac{\text{maximum area of XFPOLY on the node}}{\text{maximum area of XGATE on the node}} = \text{ratio}$$

The maximum node in the figure has a ratio of 1.6.



If you use only one parameter file, the `COMPUTE MAX` function calculates the value for each separate area. For example, the following `COMPUTE` function reports the maximum separate area for each node on the *XFPOLY* layer.

```
; <functions omitted>
PARSET ANT AREA
; <functions omitted>
LEXTRACT ANT FPOLY BY NODE XFPOLY
COMPUTE  MAX XFPOLY ANTENNA
```

Using the SUM Option

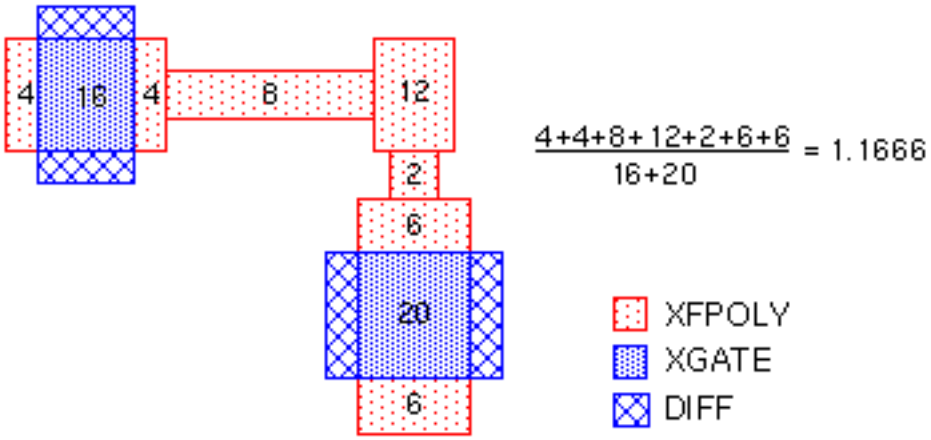
For the following check

```
COMPUTE SUM XFPOLY XGATE ANTENNA
```

the SUM option uses the following equation:

$$\frac{\text{sum of area of XFPOLY on the node}}{\text{sum of area of XGATE on the node}} = \text{ratio}$$

The node in the figure has a ratio of 1.1666.



If you use only one parameter file, the COMPUTE SUM function calculates the value for each separate area. For example, the following COMPUTE function reports the sum of all *XFPOLY* areas for each node.

```
; <functions omitted>
PARSET ANT AREA
; <functions omitted>
LEXTRACT ANT FPOLY BY NODE XFPOLY
COMPUTE  SUM XFPOLY ANTENNA
```

Using the MINGATE Option

You must create three input parameter files for the MINGATE option. For example

```
OR    NGATE PGATE GATE
NOT   POLY  GATE  FPOLY
STAMP PGATE BY    POLY
STAMP NGATE BY    POLY
STAMP FPOLY BY    POLY
LEXTRACT ANT NGATE BY NODE XNGATE
LEXTRACT ANT PGATE BY NODE XPGATE
LEXTRACT ANT FPOLY BY NODE XFPOLY
```

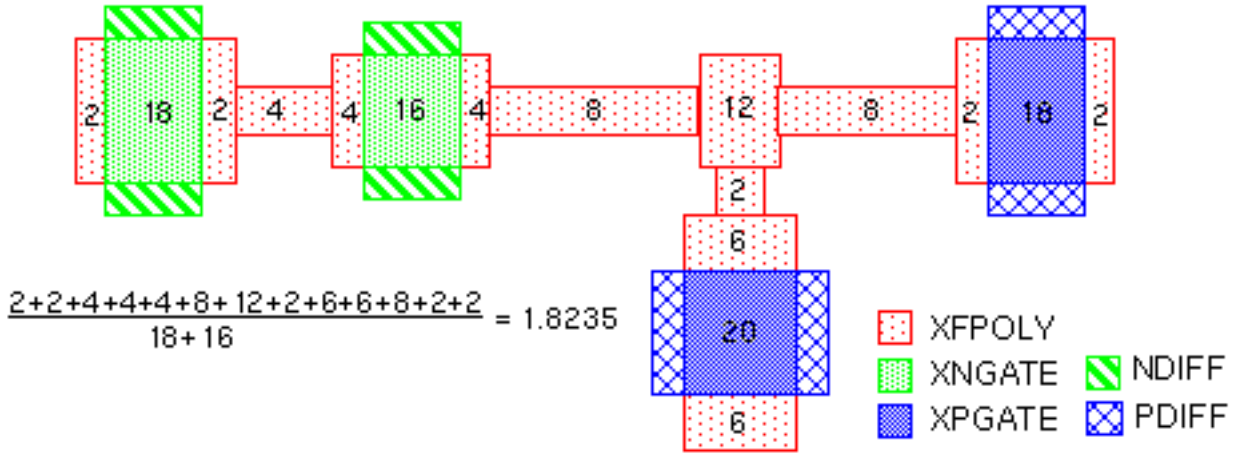
For the following check

```
COMPUTE MINGATE XFPOLY XNGATE XPGATE ANTENNA
```

the MINGATE option uses the following equation:

$$\frac{\text{sum of area of XFPOLY on the node}}{\text{the smaller of (the sum of the area of XNGATE) or (the sum of the area of XPGATE) on the same node}} = \text{ratio}$$

The node in the figure has a ratio of 1.8235.0



$$\frac{2+2+4+4+4+8+12+2+6+6+8+2+2}{18+16} = 1.8235$$

Extracting Area Ratios Using the CALCULATE Function

The CALCULATE function performs the same MIN, MAX, and SUM operations that COMPUTE does. This function also lets you specify detailed equations for ratio checking. The following example shows how you can extract several parameters, such as area and perimeter, and use them selectively with options such as MAX and SUM.

```
; <functions omitted>
PARSET ANT AREA PERI
; <functions omitted>
LEXTRACT ANT METAL1 BY NODE XMET1
LEXTRACT ANT METAL2 BY NODE XMET2
LEXTRACT ANT PGATE  BY NODE XPGATE
LEXTRACT ANT NGATE  BY NODE XNGATE
;
CALCULATE RATIO = (MAX.XNGATE.PERI + 0.1*SUM.XMET1.AREA + 0.5*MAX.XMET2.PERI)/
MAX.XPGATE.AREA
```

For information about the CALCULATE function, see the "CALCULATE" section in the "Operation Block Commands" chapter of the *Dracula Reference.*

Sample Rules File

You can use the rules file in this section as a guide for writing antenna checks. This rules file checks poly to gate ratios as well as cumulative field poly-metal1 to gate ratios using the CALCULATE command.

```
*DESCRIPTION
;
; <functions omitted>
;
; Parameters for antenna checking
PARSET ANTE AREA
;
;<functions omitted>
;
*END

;
*INPUT-LAYER
 pwell         =  1
 active        =  2
 poly          =  3
 pimplnt       =  4
 contact       =  5
 metal1        =  6
 via           =  7
 metal2        =  8
 TEXT          = 30
 SUBSTRATE     = bulk 63
 CONNECT-LAYER = nsub pwell nsd psd poly m1ant m2ant
*END

*OPERATION
;Create recognition layer and s/d terminals for MOS transistors
 AND pimplnt active pplus
 NOT active pplus nplus
 ANDNOT pplus poly pgate psd
 ANDNOT nplus poly ngate nsd
 NOT bulk pwell nsub
 AND psd pwell ptap
 AND nsd nsub ntap

;Create layers for antenna checking
 OR     ngate    pgate     gate
 NOT    poly     gate      fpoly
 AND    contact  poly      polycon
 SELECT metal1   OVERLAP   polycon  m1ant
 SELECT metal2   OVERLAP   m1ant    m2ant

;Define connectivity of the circuit
 CONNECT  m2ant  m1ant BY via
 CONNECT  m1ant  poly  BY contact
 CONNECT  m1ant  nsd   BY contact
 CONNECT  m1ant  psd   BY contact
 SCONNECT nsd    nsub  BY ntap
 SCONNECT psd    pwell BY ptap

;Pass node information for antenna checking
 STAMP    gate   BY    poly
```

```
;
*BREAK ANTE

;Extract parameters for antenna checking
 LEXTRACT ANTE m2ant BY NODE xm2ant
 LEXTRACT ANTE m1ant BY NODE xm1ant
 LEXTRACT ANTE gate  BY NODE xgate

;Calculate the ratio of the sum of the metal1 to the sum of all gate
;areas
 CALCULATE parm1 = xm1ant.area / xgate.area

;Calculate the ratio of the sum of the metal2 to the sum of all gate
;areas
 CALCULATE parm2 = xm2ant.area / xgate.area

;Calculate the cumulative ratio of the sum of all the metal to the
;sum of all gate areas
 CALCULATE carme = (xm1ant.area+xm2ant.area) / xgate.area

;If the ratio of metal1 areas to gate areas is more than 2.5,
;output the gate geometries
 CHKPAR PAR parm1 gate GT 2.5 eme1 OUTPUT e1 63

;If the ratio of metal2 areas to gate areas is more than 2.5,
;output the gate geometries
 CHKPAR PAR parm2 gate GT 2.5 eme2 OUTPUT e2 63

;If the cumulative ratio of all the metal areas to gate areas is more
;than 2.5, output the gate geometries
    CHKPAR PAR carme gate GT 2.5 emet OUTPUT et 63

*END
```

Writing antenna rules using the drcAntenna macro

The drcAntenna macro was recently introduced into Dracula to make the task of writing antenna checks easier. In a nutshell, the macro is of the form:

```
drcAntenna(
    gate( gate layer list )
    antenna( antenna list )
    diff( diffusion layer list )
    parset( parset list )
    check( antenna check list )
)
```

For a complete explanation of the macro syntax, please refer to the "drcAntenna" section in the "Operation Block Commands" chapter of the *Dracula Reference*.

The following example rewrites the previous Sample Rules File section using the drcAntenna macro:

```
*DESCRIPTION
;<functions omitted>
*END
```

```
*INPUT-LAYER
 pwell         =  1
 active        =  2
 poly          =  3
 pimplnt       =  4
 contact       =  5
 metal1        =  6
 via           =  7
 metal2        =  8
 TEXT          = 30
 SUBSTRATE     = bulk 63
 CONNECT-LAYER = nsub pwell nsd psd poly metal1 metal2
*END


*OPERATION
;Create recognition layer and s/d terminals for MOS transistors
 AND pimplnt active pplus
 NOT active pplus nplus
 ANDNOT pplus poly pgate psd
 ANDNOT nplus poly ngate nsd
 NOT bulk pwell nsub
 AND psd pwell ptap
 AND nsd nsub ntap

;Create layers for antenna checking
 OR ngate pgate gate

;Define connectivity of the circuit
 CONNECT metal2 metal1 BY via
 CONNECT metal1 poly BY contact
 CONNECT metal1 nsd BY contact
 CONNECT metal1 psd BY contact
 SCONNECT nsd nsub BY ntap
 SCONNECT psd pwell BY ptap


*BREAK ANTE
 drcAntenna(
  gate   ( gate poly )
  antenna( metal1 metal2 )
  diff   ( psd nsd )
  parset ( ANTE AREA )
  check  ( ( metal1 (CALC parme1 = metal1.AREA / gate.AREA)
                    (CHKPAR PAR parme1 gate GT 2.5 eme1 OUTPUT e1 63)
           ) ;end of metal1 check
         ( metal2 (CALC parme2 = metal2.AREA / gate.AREA)
                    (CALC carme  = parme1 + parme2)
                    (CHKPAR PAR parme2 gate GT 2.5 eme2 OUTPUT e2 63)
                    (CHKPAR PAR carme gate GT 2.5 emet OUTPUT et 63)
           ) ;end of metal2 antenna check
         ) ;end of check function
 ) ;end of drcAntenna

*END
```

Writing In-the-Direction-Of Checks

You might need to check whether one layer encloses another layer by a certain dimension, but by a different dimension in the direction of a third layer. For example, poly must enclose contacts by 2 microns, but by 2.5 microns in the direction of metal (current).

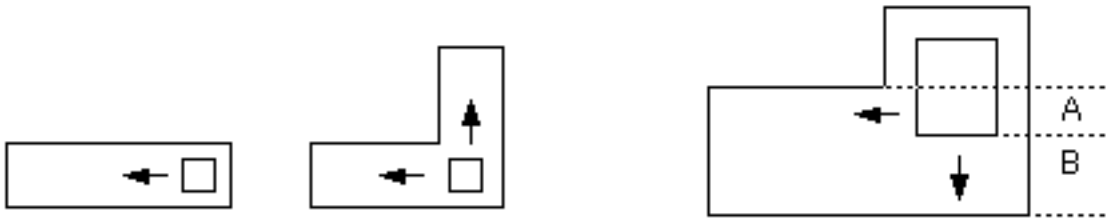
You might also need to write a check for two layers, such as metal and contact, to check spacing in the direction of current.

Choosing the Direction

To write in-the-direction-of checks, you must first determine what the direction is. The following figure shows some possible in-the-direction-of cases.

The two cases on the left are straightforward, because the direction is toward the mass of the conductor layer. For the case on the right, you need to decide if the *A* or *B* distance warrants an in-the-direction-of check.

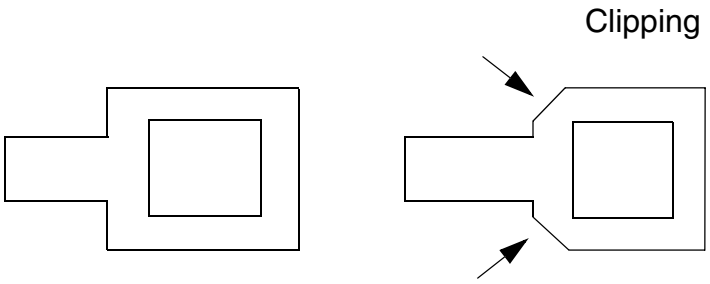
- If *A* is greater than a certain value, left becomes a valid direction.
- If *B* is greater than a certain value, down becomes a valid direction.



Once you decide what the *A* and *B* values are, you can use them as the minimum spacing values in your in-the-direction-of checks.

Clipping Geometries

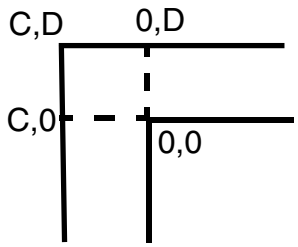
Some technologies allow clipping, while others do not. The shape on the left in the following figure is not clipped. The shape on the right is clipped.



If your technology does not allow clipping, you will need to write your basic check for minimum enclosure using the `ENC[S]` option.

Enclosure checks for designs that don't allow clipping use the `[S]` option with the `ENC` function. The `[S]` option takes two measurements for every point between the two geometries. If both these measurements are less than the spacing you specify, Dracula reports an error.

When taking the measurements for the `[S]` option, Dracula checks the x,y value for each dimension. For example, point C,0 in the following figure measures C on the x axis and 0 on the y axis. Point 0,D measures 0 on the x axis and B on the y axis. If your enclosure value is 3, and both C and D are less than 3, Dracula reports an error.



If both C and D are less than the enclosure value, Dracula reports an error.

For examples using the `[S]` option, see "Writing the Basic Check" section.

Writing the Basic Check

Always check the minimum distance around a contact first. For example, write the global check that metal encloses contacts by at least 1 micron in any direction. For example

```
; Minimum contact enclosure check, clipping allowed.
ENC CONTACT METAL1 LT 1 OUTPUT ERR 01
;
```

```
;Minimum contact enclosure check, clipping not allowed.
ENC[S] CONTACT METAL1 LT 1 OUTPUT ERR 02
;
; Minimum contact enclosure check, partial clipping allowed.
ENC[S] CONTACT METAL1 LT 1 CORNER-EDGE = .5 OUTPUT ERR 03
```

See Clipping Geometries for information about the ENC[S] function.

For Designs Where Clipping Is Allowed

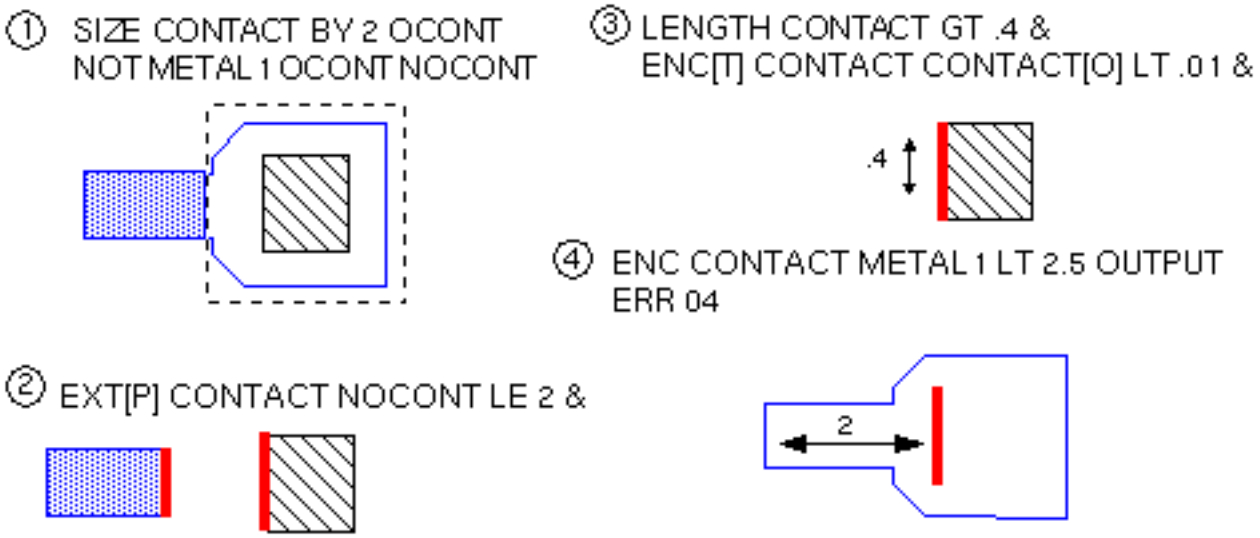
If clipping is allowed in your design, and your contacts are all squares of the same size, you can use an in-the-direction-of check like this:

```
; In-The-Direction-Of Check.
; The minimum length of a portion of a contact edge at
; which a direction check is required = .4.
;
SIZE CONTACT BY 2 OCONT
NOT METAL1 OCONT NOCONT
EXT[P] CONTACT NOCONT LE 2 &   ; get in-direction-of edges
LENGTH CONTACT GT .4 &         ; find those of min. length
ENC[T] CONTACT CONTACT[O] LT .01 & ; select contact edges
ENC CONTACT METAL1 LT 2.5 OUTPUT ERR 04
```

Note: The value used for the SIZE and EXT functions is the *B* value, described in Choosing the Direction.

Caution

The rules in this section are conjunctive rules. These rules use the whole edge, not just the portion of the edge that is in-the-direction-of. To avoid false errors using conjunctive rules, see Writing Edge Checks***.***



For Designs Where Clipping Is Not Allowed

If clipping is not allowed in your design, use an in-the-direction-of check like this:

```
; In-The-Direction-Of Check
; The minimum length of a portion of a contact edge at
; which a direction check is required = .4,
; clipping not allowed.
;
SIZE CONTACT BY 2 OCONT
NOT METAL1 OCONT NOCONT
EXT[P] CONTACT NOCONT LE 2 &
LENGTH CONTACT GT .4 &
ENC[T] CONTACT CONTACT[O] LT .01 &
ENC[S] CONTACT METAL1 LT 2.5 OUTPUT ERR 05
```

Note: The value used for the `SIZE` and `EXT` functions is the *B* value, described in Choosing the Direction.

This check is identical to the check where clipping is allowed, except that it uses `ENC[S]` rather than `ENC`. See the "Clipping Geometries" section for information about the `ENC[S]` function.

The rules in this section are conjunctive rules. These rules use the whole edge, not just the portion of the edge that is in-the-direction-of. To avoid false errors using conjunctive rules, see the section, "Writing Edge Checks***."***

Writing Exact Contact Size Checks

You might need to check that all geometries are an exact size, such as ensuring that all contacts are exactly 2x2 or 1x4. The following rules file shows the technique you can use to check for exact contact size checks.

```
;
; <functions omitted>
;
*INPUT-LAYERS
contact = contact
*END
;
*OPERATION
; Contacts can be 1x4, 1x2 or 4x4 only.
; All others are violations.
;
; The WIDTH function will flag any acute angles.
;
; For 1x4 contacts:
;
;   Find contacts with 1x4 area
AREA contact EQ 4 cont1x4
;
;   Subtract these contacts from the other contacts
NOT contact cont1x4 contx
;
;   Do width check for 1x4 contact, only checking those
;   that fit the area constraints.
WIDTH  cont1x4 LT 4 &
LENGTH cont1x4 LT 4 OUTPUT err1 63
;
; For 1x2 contacts, follow the same procedure using the
; remaining contacts available.
;
AREA   contx   EQ 2 cont1x2
NOT    contx   cont1x2 ocnts
WIDTH  cont1x2 LT 2 &
LENGTH cont1x2 LT 2 OUTPUT err2 63
;
; For 4x4 contacts, using the contacts that are left.
;
AREA   ocnts EQ 16 cont4x4
WIDTH  cont4x4 LT 4 &
LENGTH cont4x4 LT 4 OUTPUT err3a 63
WIDTH  cont4x4 LT 4 &
LENGTH cont4x4 GT 4 OUTPUT err3b 63
;
```

```
; Then output the remaining contacts that did not fit any
; of the previous constraints.
;
OR  cont1x4 cont1x2 tmp
OR  tmp     cont4x4 okconts
NOT contact okconts badcont OUTPUT err4 63
*END
```

Writing Metal Reflection/ Crosstalk Checks

You can find a detailed discussion of metal-over-poly and metal-over-diffusion reflection checks in the "DRC Example" section in the "Checking Design Rules (DRC)" chapter of the *Dracula Reference.*

Writing Electromigration Checks

To locate areas of potential electromigration, you can use the following techniques in Dracula.

- Gate approximation

  You can use gate dimensions to determine the potential current, then use other layout dimensions to determine how much current the contacts can handle.

- Fuse extraction

  You can use Dracula to extract a fuse network that represents areas of potential electromigration, then use your simulator to locate those areas.

Using Gate Approximation

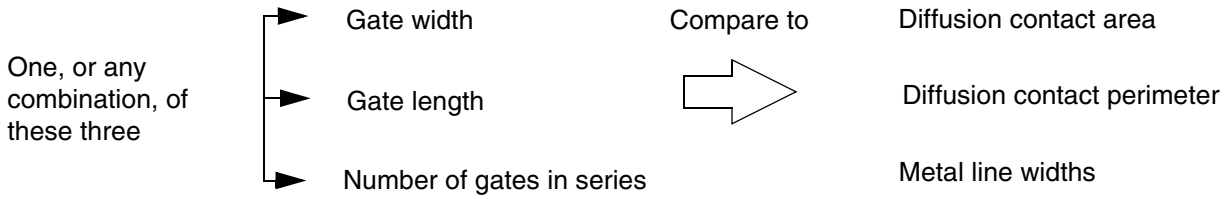
The gate approximation method is based on these concepts:

- You can use gate dimensions to predict peak current.
- You can use diffusion contact dimensions to predict the amount of current the contacts can handle.

To locate potential electromigration, you compare peak currents from gate dimensions to the amount of current diffusion contacts can handle, based on contact dimensions. Electromigration occurs when drive current is greater than the current the diffusion contacts can handle.

The advantage to this method is that you can use it very early in the design cycle. However, it works only on localized problems. Also, this method can be used to predict only whether there is a potential for an electromigration problem.

You can look for possible electromigration problems by comparing one or more of the values on the left to one of the values on the right.



You can find examples of these techniques in the sections that follow.

The gate approximation method uses the `COMPUTE SUM` function. For a description of this function, see Extracting Area Ratios Using the COMPUTE Function.

Comparing Gate Width to Contact Area

The following rules file uses these elements in its electromigration equation:

- Drive current

  *gate_width * current_value1*

- Current that the diffusion contacts can handle

  *contact_area * current_value2*

The current values are values you supply, based on your process technology, with the `CHKPAR` function.

You look for possible electromigration problems in areas where

(*gate_width*current_value1*) > (*contact_area*current_value2*)

which is the same as

(*gate_width*/*contact_area*) > (*current_value2*/*current_value1*)

The rules file that follows extracts the gate width and contact area, computes the area ratio of gate width to contact area, and reports the areas where this ratio indicates that electromigration might be a problem.

```
*DESCRIPTION
;
; <functions omitted>
;
PARSET TST AREA
*END
*INPUT-LAYER
DIFF = DIFF
POLY = POLY
```

```
CONT = CON
;
CONNECT-LAYER = DIFCON SD GATEDG
*END
*OPERATION
NOT DIFF POLY SD
;
; Find source/drain contacts
AND CONT SD DIFCON
;
; Find gate/SD edges (gate width)
EXT[TR] POLY SD LT .1 GATEDG
;
; Set up contact layers
AND GATEDG SD GSD
;
; Connect gate/SD edges with SD contacts
CONNECT GATEDG SD BY GSD
CONNECT SD DIFCON BY CONT
;


; LEXTRACT doesn't work unless there is at least one
; ELEMENT function in the rules file.
; These functions generate an empty device.
AND DIFCON GATEDG EMPTY
ELEMENT CAP EMPTY DIFCON GATEDG
PARAMETER CAP 1
;
; Extract GATEDG area, keeping the node information
LEXTRACT TST GATEDG BY NODE EDGEAR
;
; Extract contact area, keeping the node information
LEXTRACT TST DIFCON BY NODE CONTAR
;
; Compute (GATEDG area)/(DIFCON area) for each node
COMPUTE SUM EDGEAR CONTAR SUMAREA
;
; Output GATE/SD edges that have a ratio of 6/10 to 1/1
; for the (GATE/SD edge)/(DIFCON area)
CHKPAR PAR SUMAREA GATEDG RA 0.6 1 OUTPUT ERR 01
*END
```

Comparing Gate Width Sorted by Length to Contact Area

The following rules file uses these elements in its electromigration equation:

- Drive current

  `(gate_width/gate_length) * current_value1`

- Current that the diffusion contacts can handle

  `contact_area * current_value2`

You look for possible electromigration problems in areas where

`(contact_area/gate_width/gate_length) > (current_value1/current_value2)`

To find areas of possible electromigration problems, you can use a rules file like the example that follows, which extracts gate width for gates of different lengths. It extracts contact area, computes the area ratio of gate width to contact area, and reports the areas where the ratio indicates that electromigration might be a problem for each gate length.

```
*DESCRIPTION
; <functions omitted>
PARSET TST AREA
*END

*INPUT-LAYER
DIFF = DIFF
POLY = POLY
CONT = CON
CONNECT-LAYER = DIFCON SD GATEDG2 GATEDG1
*END


*OPERATION
;
; <functions omitted>
;
NOT DIFF POLY SD
AND DIFF POLY GATE
;
;-----------------------------------------------
; Find gates of different sizes
EXT[R] SD LE 1 GLE1
EXT[R] SD RANGE 1 2.1 G1TO2
; Gates of length less than or equal to 1
AND GATE GLE1 GATE1
; Gates of length greater than 1 but less than
; or equal to 2
AND GATE G1TO2 GATE2
;-----------------------------------------------
;


; Find gate/SD edges
EXT[TR] GATE1 SD LT .1 GATEDG1
EXT[TR] GATE2 SD LT .1 GATEDG2
;
; Set up contact layers
AND GATEDG1 SD GA1
AND GATEDG2 SD GA2
;
; Find source/drain contacts
AND CONT SD DIFCON
;
; Connect gate/SD edges with SD contacts
CONNECT GATEDG1 SD BY GA1
CONNECT GATEDG2 SD BY GA2
CONNECT SD DIFCON BY CONT
;
; LEXTRACT doesn’t work unless there is at least one
; ELEMENT function in the rules file.
; These functions generate an empty device.
AND DIFCON GATEDG1 EMPTY
ELEMENT CAP EMPTY DIFCON GATEDG1
PARAMETER CAP 1
;
```

```
; Extract GATEDG1 area, keeping the node information
LEXTRACT TST GATEDG1 BY NODE EDGARE1
; Extract GATEDG2 area, keeping the node information
LEXTRACT TST GATEDG2 BY NODE EDGARE2
; Extract contact area, keeping the node information
LEXTRACT TST DIFCON BY NODE CONTARE
;
; Compute (GATEDG area)/(DIFCON area) for each node
COMPUTE SUM EDGARE1 CONTARE SUMAR1
COMPUTE SUM EDGARE2 CONTARE SUMAR2
;
; Output gate/SD edges that have a ratio of 6/10 to
; 1/1 for (gate/SD edge)/(DIFCON area)
; For gate lengths LE 1
CHKPAR PAR SUMAR1 GATEDG1 RA .6 1 OUTPUT ERR 01
; For gate lengths GT 1 & LE 2
CHKPAR PAR SUMAR2 GATEDG2 RA .6 1 OUTPUT ERR 02
;
*END
```

Comparing Gate Width Sorted by Number of Gates in Series to Contact Area

The following rules file uses these elements in its electromigration equation:

- Drive current

  *(gate_width/number_of_gates) * current_value1*

- Current that the diffusion contacts can handle

  *contact_area * current_value2*

You look for possible electromigration problems in areas where

(contact_area/gate_width/number_of_gates) > (current_value1/current_value2)

If you want to sort the gate width information according to whether the gate is single or serial, you can use the rules file in Comparing Gate Width Sorted by Length to Contact Area. You need to replace some lines in this rules file to sort gate widths by single or serial gates.

Replace these lines:

```
;-----------------------------------------------
; Find gates of different sizes
EXT[R] SD LE 1 GLE1
EXT[R] SD RANGE 1 2.1 G1TO2
; Gates of length less than or equal to 1
AND GATE GLE1 GATE1
; Gates of length >1 but less than or equal to 2
AND GATE G1TO2 GATE2
;-----------------------------------------------
```

with these lines:

```
;-----------------------------------------------
; Find diffusion that contains only one gate
SELECT DIFF ENCLOSE[1:1] GATE DIFF1
```

```
; Find diffusion that contains two devices in series
SELECT DIFF ENCLOSE[2:2] GATE DIFF2
; Find gates that are not in series
AND DIFF1 GATE GATE1
; Find gates that are in series with only one other device
AND DIFF2 GATE GATE2
;-----------------------------------------------
```

Comparing Gate Width to Contact Perimeter

The following rules file uses these elements in its electromigration equation:

- Drive current

  *gate_width * current_value1*

- The current that the diffusion contacts can handle

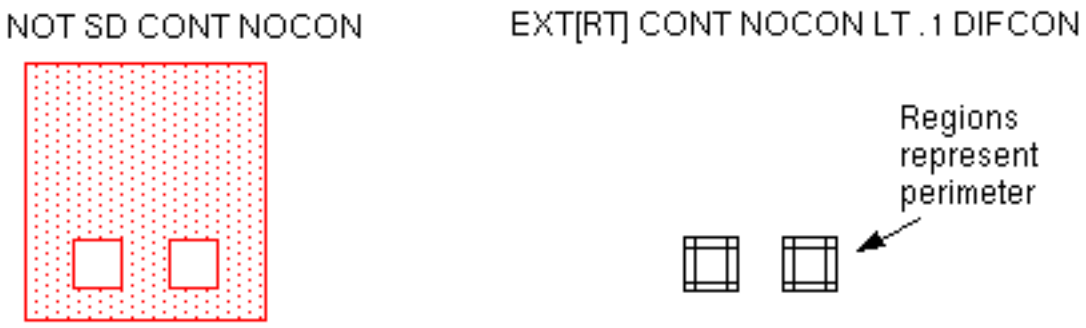
  *contact_perimeter * current_value2*

You look for possible electromigration problems in areas where

(gate_width/contact_perimeter) > (current_value2/current_value1)

To find areas of possible electromigration problems, you can use a rules file like the example that follows, which extracts gate width and contact perimeter. It computes the area ratio of gate width to contact perimeter, and reports the areas where this ratio indicates that electromigration might be a problem.

Although the parameter set in this rules file extracts only area, the perimeter of the contacts is obtained using the EXT function. First, contacts are removed from the source/drain diffusion to create the NOCON layer, shown in the following figure. Then an EXT function using the Region and Touch options creates regions around the edges where the contacts touch the diffusion. These regions, which are one resolution unit wide, represent the perimeter of the contact.




```
*DESCRIPTION
;
```

```
; <functions omitted>
;
PARSET TST AREA
*END
*INPUT-LAYER
DIFF = DIFF
POLY = POLY
CONT = CON
CONNECT-LAYER = DIFCON SD GATEDG
*END
*OPERATION
NOT DIFF POLY SD
;
; Find source/drain contacts
NOT SD CONT NOCON
EXT[RT] CONT NOCON LT .1 DIFCON
;
; Find gate/SD edges (gate width)
EXT[TR] POLY SD LT .1 GATEDG
;
; Set up contact layers
AND GATEDG SD GSD
AND DIFCON SD SDCON
;
; Connect gate/SD edges with SD contacts
CONNECT SD DIFCON BY SDCON
;
; LEXTRACT doesn’t work unless there is at least one
; ELEMENT function in the rules file.
; These functions generate an empty device.
AND DIFCON GATEDG EMPTY
ELEMENT CAP EMPTY DIFCON GATEDG
PARAMETER CAP 1
;
; Extract GATEDG area, keeping the node information
LEXTRACT TST GATEDG BY NODE EDGEAR
;
; Extract contact perimeter, keeping the node information
LEXTRACT TST DIFCON BY NODE CONTPR
;
; Compute (GATEDG area)/(DIFCON perimeter) for each node
COMPUTE SUM EDGEAR CONTPR SUMAREA
;
; Output GATE/SD edges that have a ratio of 2 to 3
; for the (GATE/SD edge)/(DIFCON perimeter)
CHKPAR PAR SUMAREA GATEDG RA 2 3 OUTPUT ERR 01
*END
```

Using Fuse Extraction

The fuse extraction method is based on the concept that electromigration problems are potential fuses. You can extract a network of these electromigration fuses and use it as input to a simulator. To predict when the fuses will pop, you use a nonstandard set of equations to develop the proper stimulus pattern and models for the simulation.

This method can be very accurate at locating electromigration effects throughout the chip. However, to find the electromigration problems, you need to have a simulator that can handle the fuse network, and you must supply the correct stimulus.

The following rules file extracts fuses as parasitic resistors and diodes.

```
*DESCRIPTION
;
; <functions omitted>
;
MODEL = MOS[N],NMOS MOS[P],PMOS DIO[N],NDIO DIO[PS],PSDIO
MODEL = DIO[PW],PWDIO DIO[BE],BEDIO DIO[BF],BFDIO
MODEL = DIO[BG],BGDIO
UNIT = CAPACITANCE,PF AREA,P PERIMETER,U RESISTENCE,U
PARSET CRAR W1 W2 W R
;
; <functions omitted>
;
*END
*INPUT-LAYER
PWELL = 1
DIFF = 2
PIMP = 4
POLY = 3
CONT = 5
MET = 6 TEXT = 30
VAPOX = 7
;
CONNECT-LAYER = BULK PWELL NSD PSD POLY MET
RCONNECT-LAYER = BULK PWELL NSD PSD POLY SMET TPMET
;
SUBSTRATE = BULK 50
;
PAD-LAYER = VAPOX
*END

*OPERATION
AND DIFF POLY GATE
NOT DIFF POLY SD
;
AND GATE PIMP PGATE
NOT GATE PIMP NGATE
;
AND SD PIMP PSD
NOT SD PIMP NSD
;
AND PSD PWELL PCONT
NOT NSD PWELL NCONT
;
CONNECT MET POLY BY CONT
CONNECT MET PSD BY CONT
CONNECT MET NSD BY CONT
CONNECT PSD PWELL BY PCONT
CONNECT NSD BULK BY NCONT
;
;Extract elements
ELEMENT MOS[P] PGATE POLY PSD BULK
ELEMENT MOS[N] NGATE POLY NSD PWELL
;
```

```
LPECHK
;
; Find the metal that carries power
SELECT MET LABEL VDD METPWR
;
; Find the metal that carries ground
SELECT MET LABEL VSS METGND
;
; Find the metal that carries power/ground
OR METPWR METGND PMET
;
; Find the metal that is not carrying power/ground
NOT MET PMET SMET
;
; Create the resistor bodies
CUT-TERM PMET CONT RPMET RSTRM MAXWIDTH=100
;
; Transfer nodal information to resistor terminals
AND MET RSTRM TPMET
;
; Build an empty layer for changing the power/ground
; contacts into fuses
NOT PWELL BULK BCC
;
RCONNECT TPMET POLY BY BCC
RCONNECT TPMET PSD BY BCC
RCONNECT TPMET NSD BY BCC
RCONNECT SMET POLY BY CONT
RCONNECT SMET PSD BY CONT
RCONNECT SMET NSD BY CONT
RCONNECT PSD PWELL BY PCONT
RCONNECT NSD BULK BY NCONT
;
; <functions omitted>
;
; P-well to N source/drain
AND NSD PWELL PWNSD
; P source/drain to substrate
NOT PSD PWELL BUPSD
;
; Set up contact resistors as diodes
AND TPMET CONT PGCONT ; contacts to power/ground
AND PGCONT POLY TPMP  ; metal/poly power/ground contacts
NOT TPMET POLY NTPMP
AND NTPMP PSD TPMS ; p/g contacts to P source/drain
AND NTPMP NSD TPMD ; p/g contacts to N source/drain
;
; <functions omitted>
;
;Extract parasitic devices
PARASITIC DIO[N]  PWNSD PWELL PWNSD
PARASITIC DIO[PS] BUPSD BUPSD BULK
PARASITIC DIO[PW] PWELL PWELL BULK
;
PARASITIC RES[M] RPMET TPMET ; power/ground metal res.
ATTRIBUTE RES[M] .01
;
; <functions omitted>
;
PARASITIC DIO[BE] TPMP TPMP POLY
PARASITIC DIO[BF] TPMS TPMS PSD
```

```
PARASITIC DIO[BG] TPMD TPMD NSD
;
; Build the netlist.
;
; <functions omitted>
;
LPESELECT[YS] DIO &
; <functions omitted>
LPESELECT[YWS] RES OUTPUT SPICE
;
*END
```

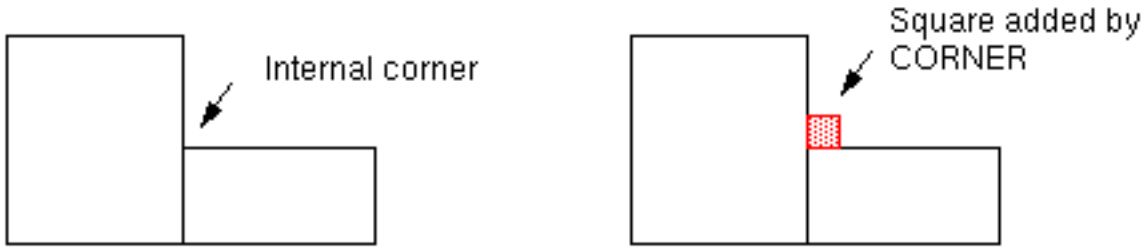
Writing Corner Checks

You can locate internal or external corners of a geometry and generate rectangles at the corner locations to flag the angles.

For example, you might want to find all internal corners of a layer and place a 2x2-unit square above this internal corner.

To do this, use the following function:

`CORNER[A] inputlayer OUTSIDE INNER CORNER-SIZE 2 outlayer`



For more information on the `CORNER` function, see “CORNER” in the “Operation Block Commands” chapter of the *Dracula Reference*.

Writing Edge Checks

You write edge checks when you want to examine only the edge of a shape, rather than the full shape. You can use two Dracula features to check edges:

- Region options

  When you need to see a portion of an edge rather than the full edge, use the region options, `R` and `R'`, of the `EXT`, `INT`, `ENC`, and `WIDTH` spacing functions.

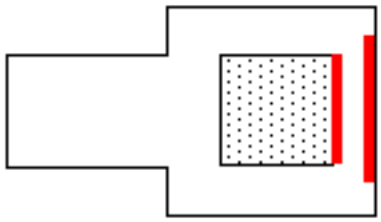
- Conjunctive rules

When you need to see the whole edge, use conjunctive rules.

Converting Error Flags into Trapezoids

When you do spacing checks, you can use the `R` and `R'` region options to convert error flags into trapezoids that Dracula can manipulate. These trapezoids represent edges that can be used in other Dracula functions.

The `R'` option builds trapezoids using error edges flagged by the spacing check. The trapezoids are one resolution unit wide and extend along the length of the error flag. The red areas in the figure show the trapezoids created by the `R'` option for an enclosure check.



You can size trapezoids created by the `WIDTH[R']`, `EXT[R']`, and `ENC[R']` functions using

- `SIZE[L]` to change the length of the trapezoid
- `SIZE[B]` to change the width of the trapezoid

Here is an example:

```
WIDTH[R'] MET1      LT 2.1  ERR1
SIZE[L]   ERR1      BY 0.6  ERR11
EXT[R']   POLY MET1 LT 2.1  ERR2
SIZE[B]   ERR2      BY 0.15 ERR22
ENC[R']   MET1 POLY LT 1.5  ERR3
SIZE[L]   ERR3      BY 0.6  ERR33
```

Using Conjunctive Rules

Unlike the `R'` option, which flags just a portion of an edge, conjunctive rules work on a complete edge.

Keep in mind that when you include the `LENGTH` function in a conjunctive rule, the input to `LENGTH` always contains only the error portions of the edge. Dracula expands the edges to full length for input to other functions in the conjunctive rule. Using edge portions for input to `LENGTH` prevents false errors in reflection rules, as shown in the following example. This

reflection rule specifies that metal lines can be 5 microns apart, as long as they are no more than 100 microns long.

```
EXT    METAL LT 5 &
LENGTH METAL GT 100 OUTPUT ERR 06
```

For this rule, the error portion must be greater than 100 microns, not the entire edge. If the entire edge was used for the length check, edges with correct spacing for a portion of the length would be flagged incorrectly.

Converting Full Edges to Edge Portions

To convert full edges to the partial edges that you obtained using the `R` or `R'` option, use the following syntax at the end of your conjunctive commands:

```
ENC[TR'] layer1 layer1[O] LT resolution output
```

Converting Edge Portions to Full Edges

To convert the partial edges that you obtained using the `R` or `R'` option to full edges, use the following syntax:

```
ENC[T] layer1 layer2 LT resolution &
```

where either *`layer1`* or *`layer2`* is generated by an `R'` check or derived from a layer generated by an `R'` check.

Measuring Edge Length

You can measure the length of a series of connected edges using the following combination of Dracula functions:

- `ENC`, `INT`, or `EXT` with the `[RT]` options
- `PLENGTH`

The value you use with the spacing check must be less than the value you specify for the `RESOLUTION` in the Description block. The following example measures all gate widths, including bent gates.

```
ENC[TR] gate poly LT .001 gatewd
PLENGTH gatewd RANGE 0 4 OUTPUT gatewd 24
```

Flagging Angles

You might need to flag corners based upon the angle that two edges make forming the corner. Dracula provides several ways to flag corners:

- `CORNER` function

  For an example of an inside corner check, see the "Writing Corner Checks" section.

- `SIZE`, `ENC`, and `NOT` functions

For more information, see the "CORNER" section in the "Operation Block Commands" chapter of the *Dracula Reference.*

You can use a combination of `SIZE`, `ENC`, and `NOT` functions as follows to create different shapes at the corners based on the corner types.

```
;
; Make a ten-resolution-unit-wide border
; around each shape
SIZE layer BY  10*resolution  SUB01
;

; Create corner markers
ENC[RC] layer SUB01 LT 11*resolution SUB02
NOT SUB01 layer SUB03
NOT SUB03 SUB02 MARKER
```

Because the markers at 45-degree angles are a different shape than markers at 90-degree angles, for example, they have different area values. You can use the `AREA` function to filter the different angles.

Defining Devices

Before you start writing device extraction rules, you might want to plan your device definitions according to these guidelines:

- Use cross-sections of devices to identify the components.

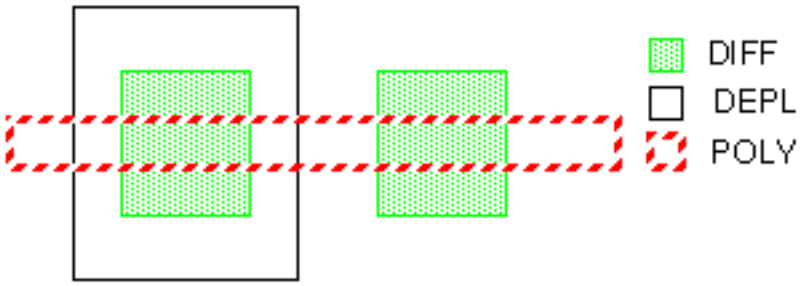
  You can visualize devices more easily with drawings. Use flat or cross-sectional drawings of the devices in your technology to identify the component layers. A sample layout containing all the device types is also useful for this purpose.

- Identify the geometric attribute that makes the device unique.

  You must create a unique recognition region for a device so that Dracula can identify that device. The geometry of the region is important, because it must touch or overlap the terminals of the device.

For example, the unique identifying feature for NMOS transistors is the depletion mask:

```
AND POLY DIFF  GBC   gates and buried contacts
NOT GBC  BCONT GATE
AND GATE DEPL  DGATE; with depletion: depletion gates
NOT GATE DGATE EGATE; no depletion: enhancement gates
; <functions omitted>
ELEMENT MOS[D] DGATE POLY SRCDRN
ELEMENT MOS[E] EGATE POLY SRCDRN
```

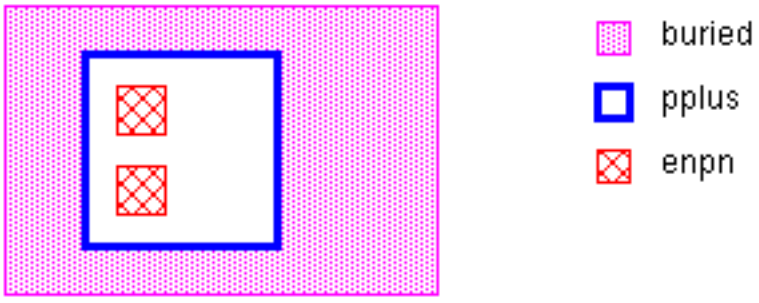


- Consider which device component to use for the recognition region.

  For bipolar technologies, for example, if you want to have a multi-emitter device recognized as a single device rather than several devices, use the emitter as the recognition region.

  The following example works for multi-emitter devices:

```
NOT sp sn cheese; shallow P NOT shallow N
SELECT sn HOLE cheese enpn
;
; <functions omitted>
;
;              device  collector   base    emitter
;              ------  ---------   ----    -------
ELEMENT BJT[N] enpn    buried      pplus   enpn
```



- Consider how Dracula will use the device definition.

For a bipolar multi-emitter device, if you want LVS to match a two-emitter device in the schematic with a two-emitter device in the layout, rather than with two one-emitter devices, define the emitter layer as the recognition layer.

- Understand the electrical interaction of all layers.

  Make sure that electrical connectivity is established on all of the terminal layers. Use CONNECT or SCONNECT to make the electrical connection, particularly for ERC applications.

ERC does not recognize connections* made with STAMP, LINK, EXPLODE, or SCONNECT with the LINK o*ption.

Connecting Your Network

When you write rules to create a network in your design, you need to use the CONNECT and CONNECT-LAYER functions. In this section, you'll learn how Dracula uses these functions to connect layers and how to avoid problems when using the connect functions.

Determining the Master Layer

When you write a CONNECT function, you can connect only from a master layer to another layer for each type of contact.

Dracula determines what the master layer is for each contact layer you specify in your rules file. Of all the layers you use in CONNECT functions for a certain contact type, Dracula selects as the master layer the layer highest in the CONNECT-LAYER function. For example, in the following rules the *CONTACT* layer connects *METAL*, *POLY*, *PSD*, and *NSD*. Of these four layers, *METAL* is the highest layer specified with CONNECT-LAYER, so Dracula chooses *METAL* as the master layer.

```
CONNECT-LAYER = NSUB PWELL NSD PSD POLY METAL
;
; <functions omitted>
;
CONNECT METAL POLY BY CONTACT
CONNECT METAL PSD BY CONTACT
CONNECT METAL NSD BY CONTACT
```

If you also include the following line, however, Dracula does not make the connection.

```
CONNECT PSD PWELL BY CONTACT
```

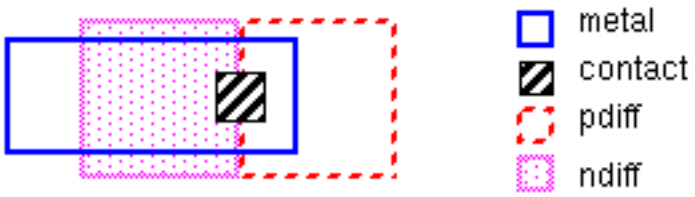
Dracula connects only from the master layer of a contact to another layer, and *PWELL* is not the master layer for *CONTACT*. Dracula does not print a message that no connections have been made between *PWELL* and *PSD*.

Connecting Multiple Layers

In Dracula, you can connect two layers only with a contact layer. You might have features on your layout that Dracula handles by dividing the contact because of this rule.

- Contact overlapping multiple layers

  If your layout has contacts that overlap multiple layers, Dracula cuts the contact. For example, the contact in the following figure is completely covered by metal, but overlaps both pdiff and ndiff. Dracula divides this contact into two contacts that connect metal to pdiff and metal to ndiff.



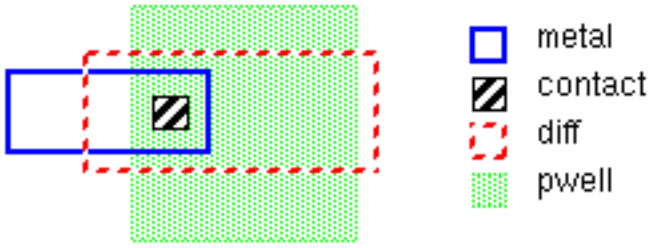
  In the next example, pdiff is not connected to anything. The contact is divided. The portion of the contact that overlaps pdiff does not overlap the master layer, which is metal, so Dracula does not make a connection.

  The contact does not connect the two diffusions. However, the nodal information is passed through the metal connection.

- Intervening layers

  If your layout has a layer intervening between two `CONNECT` layers, Dracula cannot connect the layers you specify. In the following example, the *DIFF* layer prevents the *METAL-PWELL* connection.

```
CONNECT-LAYER = PWELL DIFF METAL
;
; <functions omitted>
;
CONNECT METAL PWELL BY CONTACT
```



Preparing To Use Dracula Interactive

If you plan to use Dracula Interactive to view the results of your Dracula job, follow these steps:

1. Put `KEEPDATA = DRACULAINTERACTIVE` in the Description block of your Dracula rules file.

   Note: You need to include this command only if you want to use `KEEPDATA` to minimize the data on the disk, but you want Dracula to keep enough information to display results in Dracula Interactive. (For more information on the `KEEPDATA` command, see the "`KEEPDATA`" command section in Chapter 11 of the *Dracula Reference*).

2. To associate comments with an output layer, put them at the end of the line that generates that layer in the Operation block.

   The comment must be on the same line that generates the layer, following a semicolon. If you need additional lines for the comment, start each one with two semicolons. For example

   ```
   ENC[R'] MET1 POLY LT 1.5  ERR3 ; metal enclosed by poly
                                  ;; LT 1.5
   ```

3. Use the `htv` function when you run LOGLVS.

   The `htv` function tells LOGLVS to generate Dracula Interactive files that let you cross-probe your design.

4. In Dracula interactive debugging tool, generate the comments file and turn on the comments display.

3

Working With 32 and 64-bit Applications

The Dracula directory structure is as follows:

- The wrappers are stored in:
  `tools/dracula/bin`
- 64-bit executables(if exist) are stored in:
  `tools/dracula/bin/64bit`
- 32-bit executables with or without 64-bit equivalent are stored in:
  `tool/dracula/bin/32bit`

By default Dracula uses 32-bit application binaries.

To enable 64-bit applications use the following UNIX environment variable

`CDS_AUTO_64BIT {ALL | NONE | "INCLUDE:list" | "EXCLUDE:list"| "list"}`

This environment variable enables you to select 32-bit or 64-bit executables using the following values:

- `ALL` Invokes all applications as 64-bit, where available. The list of executables available is in your `install_dir/bin/64bit`
- `NONE` Invokes all applications as 32-bit.
- `"INCLUDE:list"` Invokes all applications in the list as 64-bit, where available.
- `"EXCLUDE:list"` Invokes all applications as 64-bit, where possible, except the applications in the list.
- `"list"` Invokes only the executables included in the list, if available, as 64-bit. "list" is a list of case-sensitive executable names delimited by a semi-colon(;) or a colon(:). Providing the list without a preceding keyword works the same as using the keyword `INCLUDE`.

Examples:

```
setenv CDS_AUTO_64BIT "pipo;layout.exe"
setenv CDS_AUTO_64BIT "EXCLUDE:si"
```

All 64-bit executables are controlled by a wrapper executable. The wrapper invokes the 32-bit or 64-bit executables depending on how the `CDS_AUTO_64BIT` environment variable is set, or whether the 64-bit executable is installed. The wrapper also adjusts the paths before invoking the 32-bit or 64-bit executables.

If you do not set the `CDS_AUTO_64BIT` environment variable then the 32-bit executable is invoked by default.

However, Dracula uses a separate environment variable called `DRACULA_AUTO_64BIT` to enable 64-bit mode. This environment variable specifies the addressing mode only for Dracula, and overrides the `ALL` or `NONE` settings of the `CDS_AUTO_64BIT` environment variable for Dracula. `DRACULA_AUTO_64BIT` recognizes the following variable values:

- `ALL` Invokes all applications as 64-bit, where available. The list of executables available is in your `install_dir/bin/64bit`
- `NONE` Invokes all applications as 32-bit.
- `"INCLUDE:list"` Invokes all applications in the list as 64-bit, where available.
- `"EXCLUDE:list"` Invokes all applications as 64-bit, where possible, except the applications in the list.
- `"list"` Invokes only the executables included in the list, if available, as 64-bit. "list" is a list of case-sensitive executable names delimited by a semi-colon(;) or a colon(:). Providing the list without a preceding keyword works the same as using the keyword `INCLUDE`.

Example:

```
setenv DRACULA_AUTO_64BIT "PDRACULA;LOGLVS"
setenv DRACULA_AUTO_64BIT "EXCLUDE:LPEPRO"
```

Important

Entry points to both the 32-bit and 64-bit addressing executables are exclusively through the wrapper programs:

- Users must not include directories containing the associated "real," non-wrapper executables in their UNIX path

- There is a consistent use model in both 32 bit and "automatic high capacity" modes (for example, the same UNIX path)
- Prevents pilot errors (for example, attempting to run a 64-bit executable in a 32-bit environment)
- Consistent with Solaris 7 32 or 64-bit model
- Requires the same licenses for a 32-bit or 64-bit operation

4

Selecting a Run Mode

This chapter discusses the following topics:

- About Run Modes
- Using Flat Mode
- Using Dracula Distributed Processing
- Using Hierarchical Mode
- Using Multilevel Mode
- Using Cell Mode
- Using Composite Mode
- Improving Performance

About Run Modes

When you are verifying your design, you might want to run Dracula in one of these modes:

- Flat mode

  Verifies every instance of every cell, and the interconnect between cells, on a single machine.

- Distributed flat mode

  Verifies every instance of every cell and the interconnect between cells. Different portions of the design are verified on different machines running at the same time.

- Hierarchical mode

  Verifies one instance of every cell and the interconnect between cells. The content of the cells is flattened to a single level.

- Multilevel hierarchical mode

Verifies one instance of every cell and the interconnect between cells. The content of the cells can have many levels.

- Cell mode

  Verifies only the contents of cells.

- Composite mode

  Verifies only the interconnect between cells.

In this chapter, you'll learn the characteristics of these run modes and get suggestions about how to select and use them to verify your design. These suggestions are general, because designs vary in size, content, structure, input format, and technology.

Different modes have different texting requirements. For details, see the texting section for each mode in this chapter. For a general discussion of texting in Dracula, see "Using Text" in the "Input-Layer Block Commands" chapter of the *Dracula Reference*.

Using Flat Mode

If you are verifying a small to medium-large layout database, use flat mode. You can use all Dracula capabilities (Dracula® design rule checker, Dracula® electrical rules checker, Dracula® flat layout versus schematic verifier, Dracula® layout parameter extractor, and Dracula® parasitic resistance extractor in flat mode. Because this mode creates a flat database, processing might take longer than a hierarchical, cell, or composite mode run.

Running Dracula in Flat Mode

To run a job in flat mode, in the Description block of your Dracula rules file you must do one of the following:

- Use `CHECK-MODE=flat`
- Omit the `CHECK-MODE` function (flat is the default)

You can run Dracula in flat mode on designs as large as your computer system can handle.

When To Use Flat Mode

Use Dracula in flat mode when

- You are new to Dracula

Start in flat mode. After you get proficient in verifying designs in flat mode, try cell, hierarchical, or composite mode runs.

- You need full functionality

  There are no limitations, methodology or design restrictions, or special requirements when you run Dracula in flat mode. You can include any or all of the Dracula capabilities (Dracula design rule checker, Dracula electrical rules checker, Dracula flat layout versus schematic verifier, Dracula layout parameter extractor, and Dracula parasitic resistance extractor in your rules file.

  For example, you can run these products and functions, which are not available in the hierarchical modes:

  - Operations that require nodal information to make connections and distinguish between nodes, such as electrical rules, antenna checks, and the `CONNECT` and `SCONNECT` functions
  - The `COVERAGE`, `RELOCATE`, `SIZE WITHIN`, and `EXPLODE` functions
  - Dracula distributed

Verifying Gate Arrays

Gate array designs consist of certain base layers overlaid by cells that create transistors, contacts, and other connections. If you run HLVS on this layout in composite mode, the base layers that lie within the cell boundaries are omitted, so devices are not formed. To correct this problem, you can do one of the following:

- Run in flat mode.
- Use the `BASE-LAYER` function in the Input-Layer block of your Dracula rules file. For more information on this function, see Using Composite Plane Geometries in Hcells in Chapter 6 of this manual.

Texting for Flat Mode

The only items you need to text for flat mode are the top-level I/Os.

Using Dracula Distributed Processing

To shorten run times, you can use distributed processing to run Dracula on multiple workstations at the same time. You supply the rules file, design, and a list of workstations to use in processing. Dracula divides the tasks among the separate workstations. Dracula automatically keeps track of which tasks can't start until other tasks are finished and which tasks need a more powerful workstation.

You can run Dracula distributed either on multiple workstations connected through an NFS network or on a multiple-CPU workstation.

When To Use Dracula Distributed Processing

You use Dracula distributed processing for

- Better throughput

  If you have multiple workstations available, you can improve performance on large, flat designs by using distributed processing.

- Flat processing

  Dracula distributed supports flat mode design rule checker, electrical rule checker, layout versus schematic verifier, layout parameter extractor, and parasitic resistance extractor jobs.

- Hierarchical processing

  Dracula distributed supports hierarchical design rule checker and composite mode layout versus schematic verifier jobs.

For more details about how to run Dracula distributed and the new enhancements in the 4.8 version, see the "Dracula Distributed Processing" section in Chapter 2 and the "MULTICPU" command and the "KEEPDATA" command sections in Chapter 11 of the *Dracula Reference*.

Using Hierarchical Mode

For design rule checks on hierarchical designs, you can maximize the advantages of your hierarchy by using Dracula in hierarchical mode. Dracula® 2-level hierarchical design rule checker (formerly called Dracula HDRC) verifies large layout databases. It takes advantage of your hierarchical design structure by recognizing repetitive cells, called *Hcells*, and performing operations on them only once, even though there might be many instances of these cells in your layout. If there are duplicate cell-to-cell and cell-to-composite relationships, these are processed only once as well. The results are reported in hierarchical form, with a single error cell for each Hcell.

Hierarchical DRC uses less disk space than flat mode. Because HDRC reports fewer errors, it simplifies debugging. Depending on the structure of your design, you might get better performance using hierarchical mode instead of flat mode.

Running Dracula in Hierarchical Mode

To run a hierarchical design rule checker, include this function in the Description block of your Dracula rules file:

```
CHECK-MODE = hier
```

Note: Hierarchical mode is supported only for hierarchical design rule checker.

What the Hierarchical Design Rule Checker Checks

HDRC checks the following:

- The contents of the Hcells
- The composite (non-Hcell) data
- Hcell-to-Hcell interfaces
- Hcell-to-composite interfaces

HDRC allows overlaps between cells. Geometries from the top level or another cell can penetrate or cross a cell. You do not need protection frames or cell boundaries around each cell. However, excessive overlaps and penetrations will slow Dracula.

When To Use the Hierarchical Design Rule Checker

Use the hierarchical design rule checker for

- Large databases

  Hierarchical design rule checker can handle large databases, for example, multimillion-transistor designs.

- Databases with repetitive structures

  Designs with large numbers of repetitive physical structures, such as memories and memory cores, are good candidates for HDRC.

- Hierarchical databases

  This mode is good for circuits containing a moderate degree of hierarchy. For highly hierarchical designs, you might consider using multilevel mode.

- Shorter run/debugging times

  Hierarchical design rule checker flags an error for an Hcell only once, rather than for all placements of the Hcell. This reduces memory requirements, error correction time, and the volume of information you need to debug your design.

To avoid performance problems, do not use HDRC on parts of your design that have these characteristics:

- ***A large number of overlapping cell instances***
- ***A large amount of over-the-cell routing***

In these cases, hierarchical processing might take longer than flat processing.

Using Multilevel Mode

If you are verifying a very large design that contains Hcells within other Hcells, you use multilevel mode. The major difference between hierarchical mode and multilevel mode is that

an Hcell in multilevel mode can contain other Hcells. You can use multilevel mode only for DRC. The goal of multilevel DRC is to support a multilevel design hierarchy to reduce verification time and disk storage requirements.

Running Dracula in Multilevel Mode

To run a DRC in multilevel mode, include this function in the Description block of your Dracula rules file:

```
CHECK-MODE = multi
```

For details about running a multilevel DRC job, see the “Hierarchical and Multilevel Hierarchical Design Rules Checker” section in Chapter 3 of the *Dracula Reference.*

Selecting Hcells for a Multilevel Run

Multilevel DRC selects suitable Hcells to check (multilevel cells with over 100,000 line segments). Because multilevel DRC is for very large designs, the criteria that Dracula uses to select multilevel Hcells is different than the criteria for selecting standard hierarchical Hcells. For a list of the multilevel Hcell selection criteria, see the “Hierarchical and Multilevel Hierarchical Design Rules Checker” section in Chapter 3 of the *Dracula Reference.*

If your circuit performs poorly in multilevel mode, you might be able to improve performance by choosing different Hcells. For more information, see Selecting Hcells Manually To Improve Performance in Chapter 6 of this manual.

When To Use Multilevel Mode

Use multilevel mode for highly hierarchical designs. Designs that have repetitive structures with many levels of hierarchy, such as large DRAMs, are good candidates for multilevel mode DRC.

Using Cell Mode

If you want to check an entire library of cells in one run instead of using one run per cell, use cell mode. Cell mode verifies a group of selected Hcells and reports errors on a cell-by-cell basis. Cell mode verifies only the cells that qualify as Hcells, not non-Hcells or interconnect between Hcells. You can use cell mode for DRC, ERC, LVS, and LPE runs.

Note: There is no cell mode PRE.

Running Dracula in Cell Mode

To run a job in cell mode, include this function in the Description block of your Dracula rules file:

```
CHECK-MODE = cell
```

You can let Dracula select Hcells automatically, or you can select them yourself. Hcells can be intermediate- or bottom-level cells in the layout hierarchy. Dracula expands all cells nested in an Hcell and brings them up to the Hcell plane.

Even though HDRC and HLVS use a similar two-level hierarchy, Hcells in HLVS are modeled differently than Hcells in HDRC. Therefore, the databases of these two checks are incompatible. You must run HDRC and HLVS separately.

When To Use Cell Mode

The usual flow for LVS/ERC/LPE is to check cells in cell mode first. Then, when all cells are verified, run a composite mode check on the cells' environment. The combination of cell- and composite-mode runs is a complete check, equivalent in completeness to a flat-mode run. Both cell mode and composite mode are required. Neither is sufficient by itself.

Do not run DRC in cell and composite modes, because cell and composite DRCs do not check Hcell-to-Hcell or Hcell-to-composite connections. Use hierarchical mode for DRCs.

Use cell mode for

- Verifying libraries

  You can place one instance of each library cell, or each new cell to add to a library, and verify the cells in cell mode.

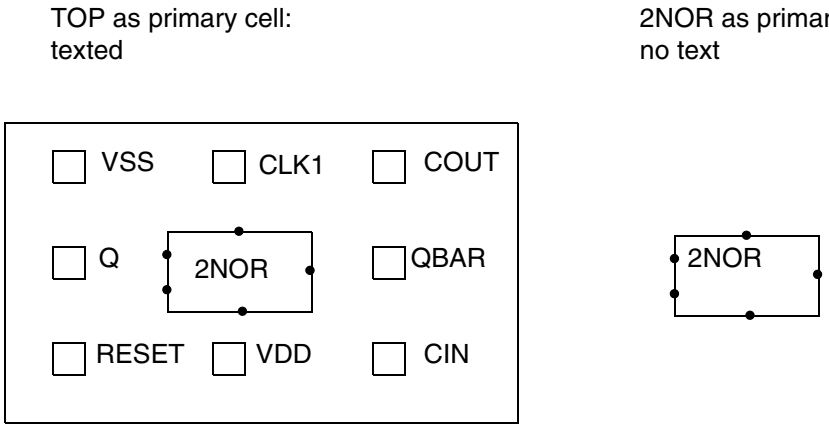
- Creating Hcells

  Use cell mode to verify the cells that you plan to use as Hcells.

- Extracting Hcell text for HLVS

In some cases, you might want to run Dracula on a cell below the top-level cell. For example, top-level cell *TOP* contains a placement of cell *2NOR*. If the Hcell text is in *TOP*, and you specify *2NOR* as the primary cell, you do not get text in cell *2NOR*.

If you want to use *2NOR* as an Hcell, you must text power, ground, and one input or output. You can then use automatic text generation to extract the text.



Run HLVS on cell *2NOR* and use the `GEN-TEXT-FILE` function in cell mode to extract Hcell text into a text file. You can then run HLVS in composite mode with the extracted text, reading the text file in with the `HEDTEXT` function. For more information, see Using GEN-TEXT To Generate Hcell Text.

Texting for Cell Mode

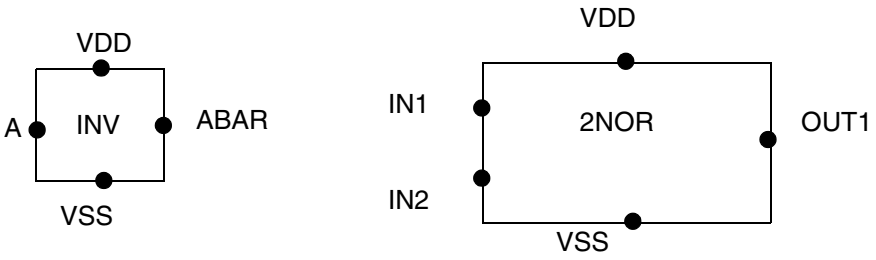
You must text the following on the layout for each Hcell in your design:

- Power
- Ground
- The minimum number of I/Os necessary to distinguish parallel circuitry

If you are running HLVS or HLPE using a netlist, the I/Os reflect the local pin names supplied in the `.SUBCKT` definition in your SPICE/CDL netlist file. For example, here is a netlist for a design:

```
*.PIN VDD VSS
.SUBCKT INV ABAR A
MP1 ABAR A VDD VDD PMOS
MN1 ABAR A VSS VSS NMOS
.ENDS
.SUBCKT 2NOR OUT1 IN1 IN2
MP1 OUT1 IN1 A   VDD PMOS
MP2 A    IN2 VDD VDD PMOS
```

```
MN1 OUT1 IN1 VSS VSS NMOS
MN2 OUT1 IN2 VSS VSS NMOS
    .ENDS
```



On the corresponding layout, these nodes must be texted:

- Inverter — *VDD*, *VSS*, *A*, and *ABAR*
- Two-Input NOR — *VDD*, *VSS*, *IN1*, *IN2*, and *OUT1*

Supplying Text from the Layout

You can use Hcell text from your layout database using the `CTEXT` function in the `INPUT-LAYER` block of your Dracula rules file:

```
*INPUT-LAYER
PWELL  = 1 CTEXT = 27 ATTACH = METAL
ACTIVE = 2 CTEXT = 33
POLY   = 4
; <functions omitted>
CTEXT = 44
CONNECT-LAYER = NSUB PWELL PSD NSD POLY METAL
*END
```

The text on layer 27 attaches to the metal geometries of your defined Hcells. Hcell text on layers 33 and 44 is attached globally, as described in the “CTEXT and ATTACH” section in Chapter 12 of the *Dracula Reference.*

Caution

Dracula versions prior to 4.3 do not read the same layer for* `TEXT` *and* `CTEXT`*. You must use different layers for Hcell text and composite text.

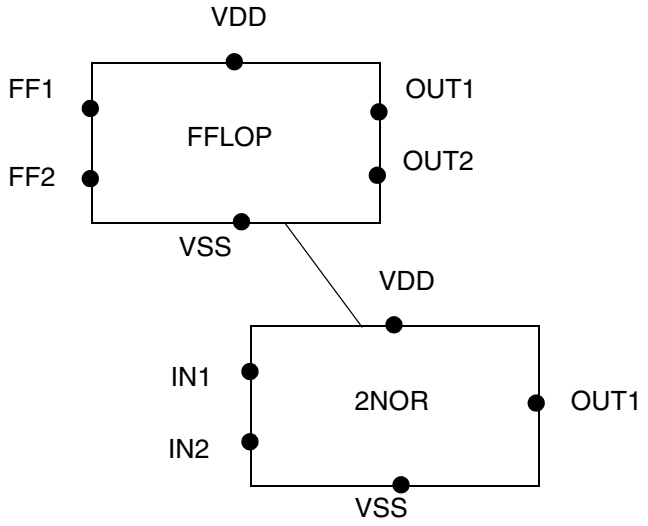
Bringing Text Up to the Hcell Level

By default, Dracula uses only database text contained within Hcells at level zero (the text must not be obscured by any hierarchy). When a cell placed within an Hcell is expanded up to the Hcell level, the text it contains is not considered part of the Hcell text.

➤ To include cell text as part of the Hcell text, add this function to the Description block of your Dracula rules file:

```
CELL-CHILD-TEXT = YES
```

If `CELL-CHILD-TEXT=YES` in the following example, text strings IN1, IN2, VDD, VSS, and OUT1 are considered part of this instance of the FFLOP Hcell.



Supplying Text from a Text File

You can supply Hcell text from an external text file. You can use this file in place of layout text or in conjunction with layout text. Any text you supply through an external file takes precedence over layout text if

- The origin of the text in the file is at the same X,Y location as the layout text
- The text in the file attaches to the same layer as the layout text

Use the `HEDTEXT` function in the Operation block of your Dracula rules file to specify an external texting file. For details, see the “HEDTEXT” section in Chapter 3 of the *Dracula Reference.*

Using GEN-TEXT To Generate Hcell Text

You can use Dracula in cell mode to generate Hcell text for a composite mode run by including the `GEN-TEXT-FILE` function in your Dracula rules file.

To use the text generation functions, you need to know how to do the following:

- Supply the minimum text required for automatic texting
- Specify texting layers

These techniques are described in the sections that follow.

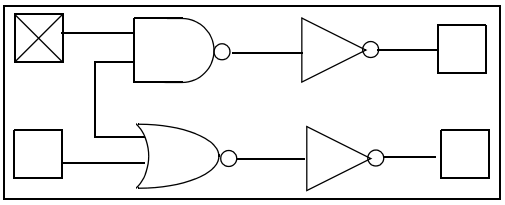
For basic information, see the GEN-TEXT Commands section in Chapter 6 of the *Dracula Reference.*

Supplying Minimum Text

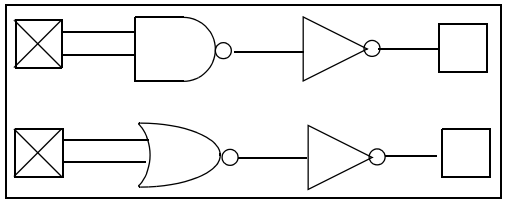
To use `GEN-TEXT`, you must supply the following minimum text for each Hcell:

- Power
- Ground
- One I/O that has a path to every element in the Hcell

You need to text only one I/O on the cell on the left in the following figure, because each I/O has a path to every element in the Hcell. You must text both I/Os on the cell on the right.



Text one I/O



Text two I/Os

If Dracula can completely verify an Hcell with this minimum text by referring to the `.SUBCKT` definition for that Hcell, all text for that Hcell is output to a file in `HEDTEXT` format. You can use this file as the Hcell text for a subsequent composite mode run.

Specifying Texting Layers

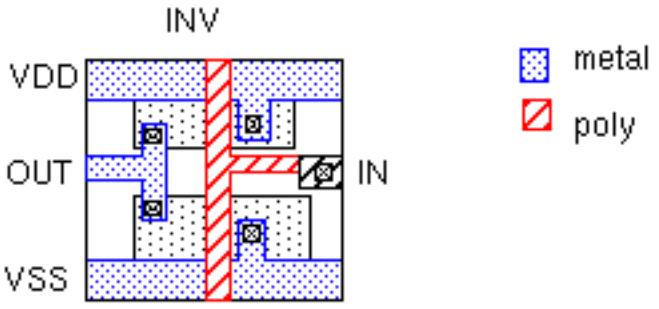
If you use the GEN-TEXT-FILE function and supply the minimum text for a composite mode run, all Dracula-generated text is attached to the layer you specified in the CTEXT function or HEDTEXT file. If a node is not on a layer specified in one of these functions, that node is not texted in the HEDTEXT file that Dracula creates.

In the following example, the only layer that text can attach to is metal:

```
*INPUT-LAYER
POLY   = 5 CTEXT = 28 ATTACH = METAL
METAL = 6
;
; <functions omitted>
;
*END
```

Without GEN-TEXT-LAYER, the default is to attach to metal only, because it is the layer you specified with ATTACH to supply the minimum text from layer 28.

- To be sure that all possible nodes are texted, supply a list of layers with the GEN-TEXT-LAYER function in the Input-Layer block.



The IN node of the INV Hcell can only be texted on poly. To attach text correctly, you need to add this line to the Input-Layer block:

```
GEN-TEXT-LAYER = POLY METAL
```

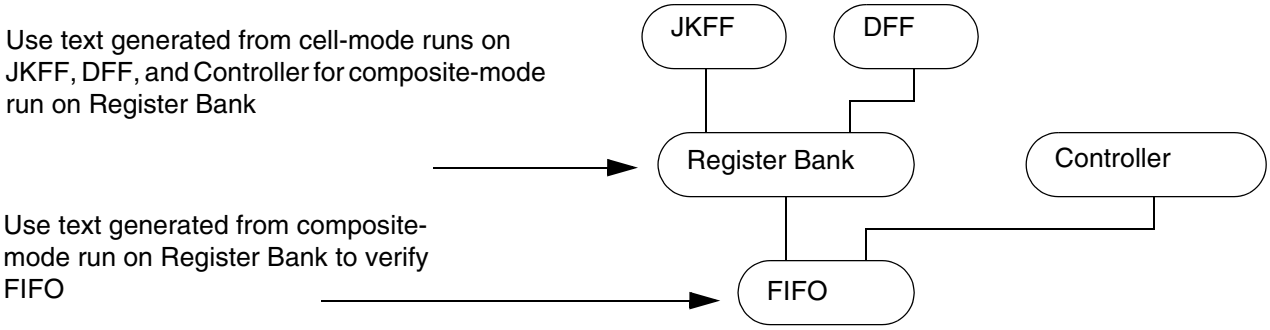
When you include this function, Dracula attaches text to metal first. If it cannot attach to metal, it attaches to poly.

Dracula attaches generated text only to the layers defined by the GEN-TEXT-LAYER function, not to layers specified in the ATTACH statement.

Texting in a Hierarchy

You can use `GEN-TEXT` functions to provide text for your design as you work up the hierarchy tree. For example, you can do a cell-mode run using `GEN-TEXT` functions on the JKFF, DFF, and Controller cells shown in the figure. Using the generated text, you can do a composite-mode run on the register bank with `GEN-TEXT` functions. You then have all the text you need to verify the FIFO cell

.



Using Composite Mode

The composite plane is the top level of the current design hierarchy. It contains all geometries not included in any Hcell boundaries.

Composite mode DRC is not a comprehensive check. It can help you find composite-level errors when you don't have time to do a full analysis of the database. However, only hierarchical or flat mode provides a complete check.

Dracula expands all cells that are neither Hcells nor cells contained in Hcells and brings them to the composite plane. The composite plane can be an intermediate cell. It does not need to be the top level of your design.

Running Dracula in Composite Mode

To run a job in composite mode, include this function in the Description block of your Dracula rules file:

```
CHECK-MODE = composite
```

Checks in Composite Mode

Use composite mode on hierarchical designs for HDRC, HERC, HLVS, HLPE, and HPRE checks. Composite mode first extracts interconnect information from cells, then processes the following checks:

- HDRC

  The interconnections between Hcells, and between Hcells and composite-level cells. Dracula doesn't check inside Hcells.

- HERC, HLVS

  Composite, Hcell, Hcell-to-Hcell, and composite-to-Hcell interconnect information.

  Composite-mode LVS does a complete electrical analysis of the contents every Hcell and verifies that there are no electrical violations between the contents and the composite data. Composite LVS doesn't LVS-check the Hcells, but it electrically analyzes every net inside every Hcell. For this reason, composite-mode LVS can take a long time, even if there are only one or two placements of each Hcell.

- HLPE

  Composite information and composite-to-cell parasitics. You can run HLPE in composite mode with the same functions as in HLVS.

- HPRE

  Designed (`ELEMENT RES`) and parasitic (`PARASITIC RES`) resistors on the composite plane, including interconnection between instances and placements.

Note: Your design cannot have nonparasitic (`ELEMENT`) devices created by overlapping Hcells.

Even though HDRC and HLVS use a similar two-level hierarchy, Hcells in HLVS are modeled differently than Hcells in HDRC. Therefore, the databases of these two checks are incompatible. You must run HDRC and HLVS separately.

When To Use Composite Mode

The usual flow for LVS/ERC/LPE is to check cells in cell mode first. Then, when all cells are verified, run a composite mode check on the cells' environment. The combination of cell- and

composite-mode runs is a complete check, equivalent in completeness to a flat-mode run. Both cell mode and composite mode are required. Neither is sufficient by itself.

Don't run DRC in cell and composite modes, because cell and composite DRCs do not check Hcell-to-Hcell or Hcell-to-composite connections. Use hierarchical mode for DRCs.

Use composite mode for

- Simplified verification analysis

  Composite mode checks a subset of your design, so it is easier to debug.

- Decreased disk space requirements

  A composite mode run requires less disk space than a flat verification run.

- Dracula-to-CDC-to-Verilog timing flow

  Use composite mode HPRE to extract RC parasitics for the interconnect, then calculate the delays from the parasitics using the Central Delay Calculator (CDC) and use the delay values in Verilog. For details, see “” on page 18.

Composite Mode and Blackbox LVS

If you have cells with internal pins, you might be able to speed up your composite LVS run by running blackbox LVS. Blackbox LVS removes the contents of Hcells, so Dracula does not read in cell data and does not consider overlapping and over-the-cell geometries. The amount of time you save depends on the density of your Hcells and their interaction with each other and the composite plane.

For information about running a blackbox LVS, see the *Dracula Blackbox LVS Verification* application note, available from your local Field Applications Engineer.

Texting for Composite Mode

You must text the following on the layout for each Hcell in your design:

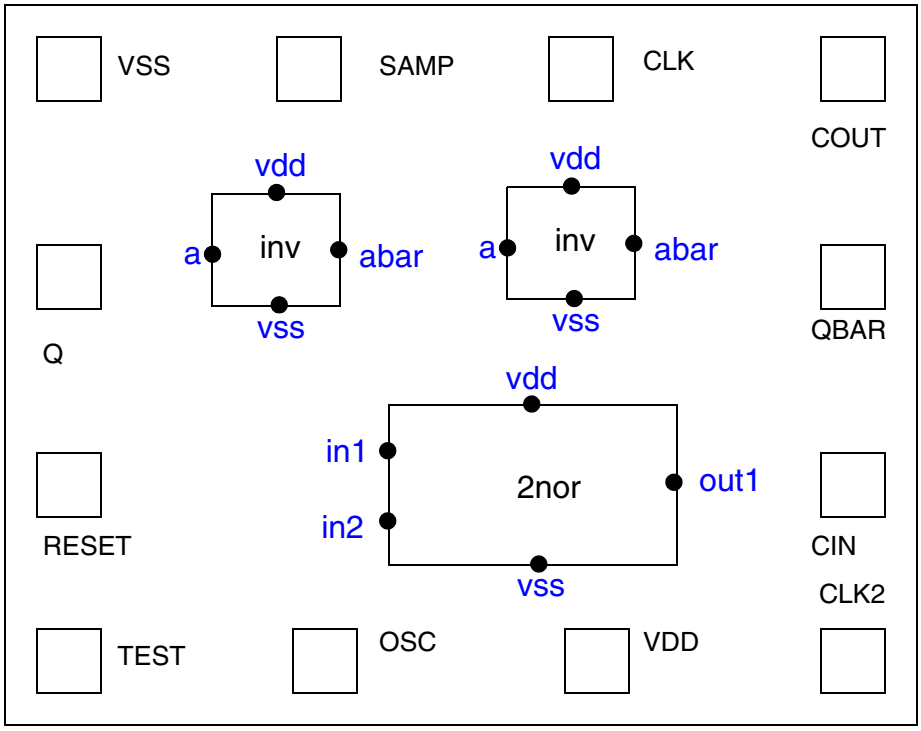
- Power
- Ground
- All I/Os

If you have verified the cell in cell mode, you already have this text.

In addition to Hcell text, you need to supply composite text:

- All top-level I/Os

The following figure shows Hcell text in regular font (blue) and composite text in red:



Supplying Text from the Layout

You extract cell and composite text from your layout database using the `TEXT` and `CTEXT` functions in the `INPUT-LAYER` block of your Dracula rules file:

```
*INPUT-LAYER
PWELL  = 1 TEXT  = 27 ATTACH = METAL
ACTIVE = 2 TEXT  = 33
POLY   = 4 CTEXT = 29 ATTACH = METAL
;
; <functions omitted>
;
CONNECT-LAYER = NSUB PWELL PSD NSD POLY METAL
*END
```

In this example, both top-level text and Hcell text are extracted from the layout. This supplies sufficient text to run Dracula in composite mode.

Dracula versions prior to 4.3 do not read the same layer for TEXT and CTEXT. You must use different layers for Hcell text and composite text.

In composite mode, you can include text for a cell inside an Hcell as part of the Hcell text. See Bringing Text Up to the Hcell Level.

Supplying Text from a Text File

You can supply composite text from an external text file. You can use this file in place of layout text or in conjunction with layout text. Any text supplied through the external file takes precedence over the layout text if

- The origin of the text in the file is at the same X,Y location as the layout text
- The text in the file attaches to the same layer as the layout text

To specify an external text file, use the `EDTEXT` function in the Operation block of your Dracula rules file. For details, see the “EDTEXT” section in Chapter 13 of the *Dracula Reference.*

Improving Performance

To improve performance, try the following techniques on your designs. Because designs vary in size, content, structure, input format, and technology, your results might vary from design to design.

Grouping Functions

You can group similar functions to reduce overhead time in loading layers, so Dracula runs faster. Similar functions that you can group are as follows:

- Booleans — `AND`, `NOT`, `OR`, `XOR`
- DRC checks — `EXT`, `ENC`, `WIDTH`, `INT`

When you group similar functions, Dracula creates fewer stages and runs faster, because it doesn't have to open and close modules as many times. Also, some modules can do multiple operations in one run. For example

Random:		**Grouped:**	
`AND`	stage 1	`AND`	stage 1
`SELECT`	stage 2	`AND`	stage 1
`AND`	stage 3	`SELECT`	stage 2
`SELECT`	stage 4	`SELECT`	stage 2

The rules on the left create four stages. Because they are not grouped, Dracula must keep re-calling the module. The rules on the right group the `AND` and `SELECT` functions, so Dracula has to call the modules only once for each group, resulting in two stages rather than four.

You can also group operations that use the same layer. Grouping similar functions for the same layer, one after the other within the Dracula rules file, reduces the time required to access the same file over and over again.

Be careful about grouping identical functions. Dracula loads all layers for identical functions at the same time, so you might run out of swap space. A* `BREAK` *function ungroups functions within a stage. Use it to separate stages if swap space is a problem.

Preallocating Swap Space

Dracula provides a function to preallocate swap space. If you don't use this function, you might need to reallocate swap space during Dracula processing.

➤ To preallocate swap space, include this function in the Description block of your Dracula rules file:

```
TRANSISTOR-NUM = value
```

where *`value`* is an estimated number of top-level transistors. The default is 100000. If swap space has to be reallocated during a Dracula run, that individual operation must be re-run, which wastes CPU time.

You can tell by looking in the Dracula log file whether your run is reallocating additional memory. Search your log file for the word `reallocate`. You might see lines like this:

```
reallocate at: EXPANX   icutex,fatt: 30     0.50           0       50000
```

or

```
reallocate at: HLOGIC   icutex,fatt: 159     0.50           0        4000
```

Note: You can also use the same function, with slightly different format, when you run LOGLVS:

```
TRANSISTOR value
```

Optimizing Rules

Dracula runs every operation you specify in your rules file. It is common for redundancies to occur within a rules file, such as two separate functions generating the same layer.

Dracula cannot automatically recognize these redundancies and optimize the rules file. To reduce processing time, you might want to read through your rules file and delete these redundancies.

X-to-Y Ratio of the Circuit

If the x range of your circuit is significantly larger than the y range, you might experience performance problems. Long horizontal circuits take longer to process because of the horizontal band scan algorithm that Dracula uses.

You can resolve this problem by rotating the circuit so that the larger dimension is in the y range. You can either place the circuit at a 90-degree rotation or use the `ROTATION=90` function in the Description block of your Dracula rules file. Be sure to consider what effect a rotation will have on your text.

5

Extracting RC Parasitics

This chapter deals with the following topics:

- About RC Extraction
- Ensuring Accuracy in RC Extraction
- Extracting Parasitic Capacitance
- Extracting Parasitic Resistance
- Extracting 2.5D MB Parasitics
- Using One Function to Extract RC Parasitics
- Extracting a Single Net
- The Dracula To RCX (dracToRcx) Interface

About RC Extraction

Parasitic resistance and capacitance (RC) between layers of an integrated circuit affect its signal speed. To calculate RC values, you define parasitics in your Dracula rules file, extract them, and use a delay calculator to calculate delays from the RC values.

In this chapter, you'll learn

- General information about achieving accuracy in RC extraction
- Basic types of RC parasitics and how to extract them

Only parasitic capacitors and resistors are described here. For more information about extracting capacitor and resistor devices, see Chapter 2, "Writing Rules for Dracula," in this manual.

The complete rules files are not included for examples in this chapter. Only the functions needed to demonstrate certain types of RC extraction are listed. When you see a "`<functions deleted>`" note in the listing, it means that lines have been omitted.

Tip

DracToRCX is an interface in Dracula that allows users to access the Assura RCX tool to perform parasitic extraction. For more information about how to use the DracToRCX interface, see "The Dracula To RCX (dracToRcx) Interface" on page 135.

Ensuring Accuracy in RC Extraction

When you supply information about your manufacturing process, Dracula uses it to extract and calculate capacitance and resistance. The accuracy of the constants you supply is an important factor in the accuracy of the values that Dracula extracts. The values you supply are as follows:

- Constant values for capacitance and resistance per unit
- Adjustments for masking and fabrication effects
- Equations for calculating resistance and capacitance values

These values are described in the sections that follow.

Providing Constants

To calculate capacitance and resistance, Dracula needs the constants you supply with the `PARASITIC CAP` and `PARASITIC RES` statements.

Caution

Make sure that the process engineers give you the most current values to use in your rules file.

For suggestions about using constants in debugging, see "Constants for Debugging Capacitance" on page 84.

Adjusting Your Design Database for Fabrication Effects

The masking and fabrication processes alter the shapes of layout geometries. The increase or decrease in size, in turn, alters resistance and capacitance of the chip. To adjust for masking and fabrication effects, follow these steps:

1. Determine the amount of distortion that masking and fabrication have on each layer.

Ask your process engineers for this information.

2. Determine whether the percentage of inaccuracy caused by this distortion is greater or less than the accuracy you want to achieve.

   For example, if the process distorts a layer by ± .01%, can you ignore the effects, or do you need to account for the distortion in the rules file?

3. If you need to adjust for distortion, determine how to implement your adjustments in the rules file.

   For example, if a layer “grows” during processing, use `SIZE` to increase its dimensions for verification. Your Field Applications Engineer (FAE) can help you with complex adjustments.

Using Equations with Flexible LPE

You might need to specify unique or detailed equations for RC extraction to reflect your manufacturing process. The Flexible LPE capability in Dracula lets you create custom equations. It consists of these functions:

- `PARSET` — lists the set of parameters that you want to extract, such as area and perimeter
- `LEXTRACT` — gives the parameter set and the layers to use in the extraction
- `EQUATION` — gives the equation to use to calculate the value you want

The following example shows a calculation for extracting capacitance. It defines a parameter set called `CPA` that lists four parameters to be extracted: perimeter (`PERI`), perimeter of the first layer overlapping the second layer (`OVPR`), area (`AREA`), and capacitance (`C`).

`C` is a reserved parameter keyword for which you supply a calculation in an `EQUATION` statement following `LEXTRACT`. You can use `PERI`, `AREA`, and `OVPR`, which are geometric primitives defined in Dracula, in your equation.

Here is an example that demonstrates Flexible LPE:

```
;
; <functions omitted>
;
PARSET CPA PERI OVPR AREA C
;
; <functions omitted>
;
LEXTRACT CPA CAP32 NME3 BY CAP[A] CAPA32 &
EQUATION C = 1 * AREA + 1 * (PERI - OVPR) + 1 * OVPR
; <functions omitted>
```

As shown in the EQUATION statement, you can specify a calculation that reflects your technology and process using predefined parameters and constants.

See the *Dracula Reference* manual for more information about:

- Parameters and Flexible LPE — The “Flexible LPE” section in the “Extracting Electrical Parameters (LPE)” chapter
- PARSET — The “PARSET” section in the “Description Block Commands” chapter
- LEXTRACT — The “LEXTRACT” section in the “Operation Block Commands” chapter
- EQUATION — The “EQUATION” section in the “Operation Block Commands” chapter

Sample rules files using Flexible LPE can be found in the following sections of this chapter:

- “Extracting Directional Sidewall Capacitance” on page 87
- “Correcting for Colinear Sidewall Capacitance” on page 90
- “Extracting Contact Resistors” on page 108
- “Correcting Sidewall Capacitance” on page 112
- “Extracting Sidewall Capacitance with Fringing Effects” on page 118

Extracting Parasitic Capacitance

The basic types of parasitic capacitance that you might want to extract are

- Overlap capacitance — a capacitor formed by the overlap of two conductive areas
- Sidewall capacitance — the capacitance between the edge of a conductive layer and the area of a conductive layer above or below it
- Fringe capacitance — the capacitance between edges on the same or different conductive layers

You might also need to include rules in your file that

- Correct capacitance calculations to remove the duplication caused by colinear edge sidewall capacitance
- Extract over-the-cell routing wire overlap capacitance with cell geometries in HLPE

➤ You extract capacitance by defining the device recognition layers and specifying the equations for calculating parasitic values in your rules file.

This section might not cover all the types of capacitance that you want to extract. However, you can use the examples in this section as general guidelines to develop similar methods for extracting parasitic capacitance from your layout.

Preparing To Extract Capacitance

To prepare for coding capacitance extraction in your Dracula rules file, do the following:

- Decide whether to output capacitance as a lump sum or cross-coupled value.
- Make sure you have power and ground connected to your circuit in a way that Dracula can understand.
- Be familiar with the Dracula functions for capacitance extraction.

These preparations are described in the sections that follow.

Selecting the Capacitance Output Format

You can use the `LPESELECT` function to output capacitances in two different ways:

- Lump sum — Capacitance between two signal nodes is duplicated for the two nodes. Then, all capacitances to a given signal node are lumped together and reported as being a capacitance to ground. When lump sum capacitance is requested, Dracula outputs *n* capacitors (per capacitor type requested), where *n* is the number of nodes.

  If your purpose is to check the capacitance loading effects on signal speed, this method is accurate enough. Here is an example of the SPICE output for node Q1B, where metal-to-metal capacitance is extracted and reported as lump sum capacitance:

  ```
  C2    Q1B  VSS  1.68510E-02PF
  ```

- Cross-coupled — When you request cross-coupled capacitance, Dracula reports the capacitance between a node and all other nodes. When cross-coupled capacitance is output, a maximum of *n*(n-1)* capacitors (per capacitor type requested) are listed, where *n* is the number of nodes.

  Because it outputs the capacitors with the nodes that form them, this method is more accurate than the lump sum method. However, the amount of output increases significantly. Here is an example of the SPICE output for the same node, Q1B, reported as cross-coupled capacitances:

  ```
  C3    Q1B  VSS  2.00000E-04PF
  C4    Q1B  X3-45  1.25101E-03PF
  C5    Q1B  X3-11  2.50000E-04PF
  C6    Q1B  Q1  6.00000E-04PF
  C7    Q1B  X3-8  4.20000E-03PF
  C8    Q1B  X3-5  1.04000E-03PF
  ```

```
C9    Q1B  X3-39  3.38778E-03PF
C10    Q1B  X3-44  4.05000E-03PF
C11    Q1B  VDD  1.87222E-03PF
```

You might try using cross-coupled output while you are debugging your rules file to see if it is easier to check where each capacitor is being formed. You can do the final report in lump sum format.

Connecting Power and Ground

To extract capacitance, you must be sure that your circuit connects to power and ground. If your circuit doesn't have power and ground, you can add them without affecting the circuit as shown in the following rules file. The following method attaches power and ground to dummy layers without affecting the circuit connectivity.

```
;
; <functions omitted>
;
*INPUT-LAYER
;
; <functions omitted>
;
METAL1 = 1
;
; <functions omitted>
;
SUBSTRATE = BULK 5
;
; <functions omitted>
;
; BL2 and BL1 are the generated VDD and GND layers
CONNECT-LAYER = BL2 BL1 METAL1
*END
;
*OPERATION
; The edtext file labels VDD and GND
; and attaches them to BL2 and BL1.
EDTEXT = exp.text
;
; <functions omitted>
;
NOT METAL1 BULK CN1  ; CN1 is an empty layer
;
; <functions omitted>
;
; The next function creates layers for
; attaching power and ground.
;
OR BULK CN1 BL1          ; for GND
OR BULK CN1 BL2          ; for VDD
;
; BL1 and BL2 must be in the CONNECT statement
; so the VDD and GND signals can be connected
; to them.
; BL1 and BL2 don't connect to anything.
;
```

```
; <functions omitted>
;
CONNECT BL1 BL2 BY CN1
;
; <functions omitted>
;
*END
```

The EDTEXT file, `exp.text` in this example, labels and attaches the VDD and GND nodes with lines like this:

```
GND X = 0 Y = 0 ATTACH = BL1
VDD X = 0 Y = 0 ATTACH = BL2
```

where coordinates (0,0) are inside the layout.

Using the Extraction Functions

This table shows you

- The steps that you follow to extract parasitic capacitors from your layout
- Which functions to use for each step
- Where to put the functions in your Dracula rules file

Capacitance Extraction Flow

Step	Function	Description	Rules file block
Define the capacitance unit	UNIT	Associates natural units with extracted parameters.	*DESCRIPTION
Specify the model name	MODEL	Associates model names of devices with circuit simulation model names.	
Create the device recognition layer	logical operations	Creates a layer that defines the area and perimeter of capacitors.	*OPERATION
Specify parasitic devices and capacitance values	PARASITIC CAP, FRINGE CAP, and ATTRIBUTE CAP	Specifies the body and terminal layers of the parasitic capacitor. Supplies values needed to calculate capacitance.	
Generate output	LEXTRACT	Extracts parameters from capacitor layers (optional).	

Step	Function	Description	Rules file block
Generate output (cont'd)	EQUATION	Used with LEXTRACT to specify the equation to use to compute the extracted capacitance. Uses the device parameters listed with the PARSET function. The LEXTRACT/EQUATION combination is called Flexible LPE (optional).	*OPERATION
	LPESELECT	Writes parasitic capacitance to an output SPICE file.	
	SPFSELECT	Writes parasitic capacitance to an output Detailed Standard Parasitic Format (DSPF) file (for HPRE jobs only).	
Generate output (cont'd)	RSPFSELECT	Writes parasitic capacitance to an output Reduced Standard Parasitic Format (RSPF) file (for HPRE jobs only; supported in Dracula version 4.2).	*OPERATION

Constants for Debugging Capacitance

The following are some techniques you might use to debug your capacitance extraction:

- You can use a constant of 1 in the ATTRIBUTE CAP statement when you first code and debug a Dracula rules file. Then, when you're sure that you're extracting the correct capacitances, you can replace 1 with accurate constants you get from the process engineers.
- Another useful technique to make your capacitance values easier to check is to extract each fringe capacitor so that the capacitance equals the width:

  ```
  ATTRIBUTE CAP[MF] 0.8 0.8
  ```

- Another method is to use "100 .01" as constants for the ATTRIBUTE CAP statement. By separating the two components by zeros, it is easy to check your results. Output

capacitances come out in *nn*00.*mm* format, where *nn* is the area component and *mm* is the perimeter.

Extracting Overlap and Sidewall Capacitance

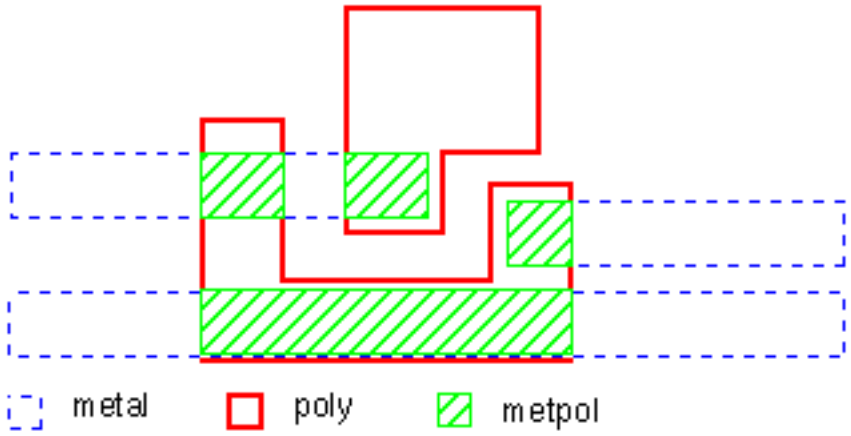
To extract overlap and sidewall capacitance, you create device recognition areas using `AND` to combine the two layers that form the capacitor. If there is an intervening third layer, you can use `NOT` to remove it. The following example removes any poly areas from the metal-to-N+ parasitic capacitance.

```
NOT MET POLY METNP ; metal with poly removed
AND METNP NSD MNSD ; metal to N+ capacitance
;
PARASITIC CAP[B] MNSD MET NSD ; metal to N+ capacitance
```

To create the parasitic capacitor and specify capacitance per area unit, use the `PARASITIC CAP` and `ATTRIBUTE CAP` functions.

The `ATTRIBUTE CAP` function specifies a capacitance value for both area (overlap) and perimeter (sidewall) calculations.

The following figure shows a capacitor recognition area created by ANDing the metal and poly layers. The resulting layer, *metpol*, is the metal-to-poly capacitor layer. The corresponding rules file follows the figure.



Rules File

```
*DESCRIPTION
;
; <functions omitted>
;
```

```
SCHEMATIC = LVSLOGIC
UNIT = CAPACITANCE,PF AREA,P
;
; <functions omitted>
;
*END
*INPUT-LAYER
;
; <functions omitted>
;
POLY = 3
METAL = 6
;
; <functions omitted>
*END
*OPERATION
;
; <functions omitted>
AND METAL POLY METPOL  metal-to-poly capacitance
;
; <functions omitted>
LVSCHK  optional ; makes schematic node
                   names available in the output
;
; <functions omitted>
PARASITIC CAP[MP] METPOL METAL POLY
ATTRIBUTE CAP[MP] 1 1
;
; <functions omitted>
LPESELECT[S] CAP GT 0.0 OUTPUT SPIC
*END
```

Note: The sidewall capacitance that you extract might be inaccurate if it includes the edges formed by cutting the layer into resistors. See "Correcting Sidewall Capacitance" on page 112.

Using the UNIT Function

The previous example used the `UNIT` function. When you use this function in your rules file, `LPESELECT` prints the unit next to the values in the SPICE file. The following example shows a portion of a SPICE file that was created without the `UNIT` function.

```
*********************************************
******   DIODE       PARAMETERS FROM : 7DIOXMER
*
D4    GND  Y  N  149.50  59.00
*
```

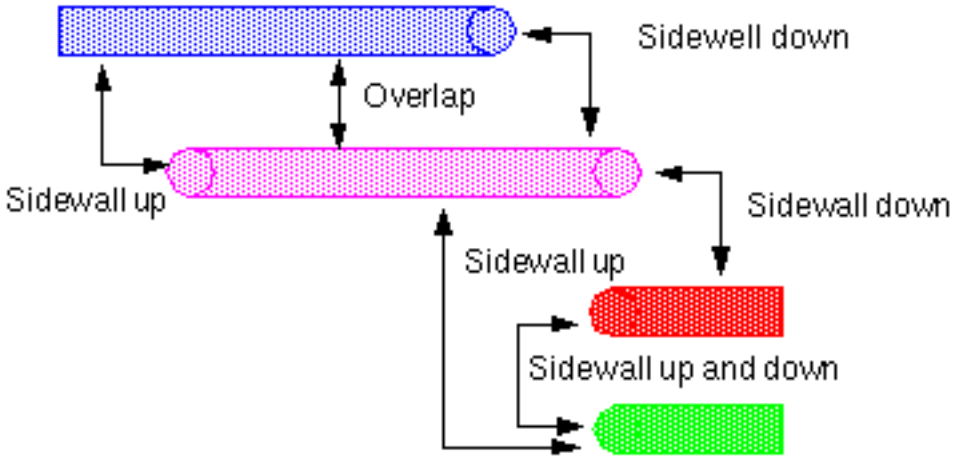
The following SPICE file was created using this `UNIT` function:

```
UNIT = AREA,P PERIMETER,U


*************************************************
******   DIODE       PARAMETERS FROM : 7DIOXMER
*
```

```
D4     GND   Y   N   149.50P   59.00U
*
```

Extracting Directional Sidewall Capacitance

You use the same functions to extract sidewall capacitance as you used to extract overlap capacitance. Where overlap capacitance is area-to-area capacitance, however, sidewall capacitance is edge-to-area capacitance. It can be directional, as shown in the following side-view layer illustration.

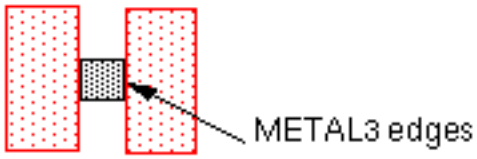


To specify perimeter, overlapping perimeter, area, and capacitance extraction, you need to define a parameter set with the `PARSET` statement. You can then use a combination of `LEXTRACT` and `EQUATION` functions to precisely extract capacitance for the directional edges.

The following example shows a directional sidewall calculation for the capacitance between *METAL3*, which is the upper layer, and the *METAL2* layer. This sidewall capacitance is extracted as shown:

```
PARSET CPA PERI OVPR AREA C
```


```
NOT BULK METAL3 NME3        AND METAL3 METAL2 CAP32
```




```
PARASITIC CAP[A] CAP32 METAL3 METAL2
ATTRIBUTE CAP[A] 1 1
LEXTRACT CPA CAP32 NME3 BY CAP[A] CAPA32 &
EQUATION C = 1 * AREA + 1 * (PERI - OVPR) + 1 * OVPR
```

The `NOT BULK METAL3` operation produces a “negative” of the *METAL3* layer, *NME3*. The overlap perimeter calculated by the `OVPR` function uses the edges that touch or overlap the *NME3* layer, in this case, the *METAL3* edges on the *CAP32* layer.

Note: The purpose of the `1` in the `EQUATION` statement is to show that you can use constant values in equations.

This equation

```
EQUATION C = 1 * AREA + 1 * (PERI - OVPR) + 1 * OVPR
```

specifies this directional sidewall capacitance:

- Sidewall-up capacitance

  The equation subtracts the `OVPR` (*METAL3*) edges from the total perimeter (`PERI`) of the *CAP32* geometry, which gives the sidewall capacitance of the *METAL2* edges to the *METAL3* area.

- Sidewall-down capacitance

The equation includes the overlap perimeter, which gives the sidewall capacitance of *METAL3* edges to the *METAL2* area.

This technique can be extended and gets more complex as more interconnect layers are included in the extraction.

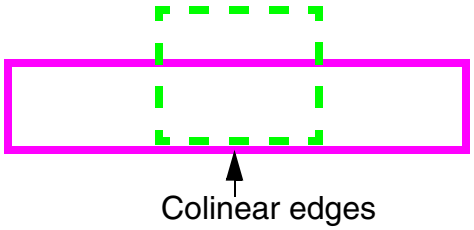
Here's an excerpt from a directional sidewall rules file dealing with three interconnect layers.

```
*DESCRIPTION
;
; <functions omitted>
;
; parameter set CPA specifies perimeter, overlap
; perimeter, and capacitance parameters
;
PARSET CPA PERI OVPR AREA C
;
; <functions omitted>
;
*END

*INPUT-LAYER
;
; <functions omitted>
;
METAL3 = 1
METAL2 = 2
METAL1 = 3
SUBSTRATE = BULK 5
;
CONNECT-LAYER = BL2 BL1 BULK METAL1 METAL2 METAL3
*END
*OPERATION
;
; <functions omitted>
;
NOT BULK METAL1 NME1
NOT BULK METAL3 NME3
;
; create capacitor recognition layers
AND METAL3 METAL2 CAP32; metal3-to-metal2 caps
;
AND METAL2 METAL1 CAP21; metal2-to-metal1 caps
;
; <functions omitted>
;
; specify parasitic capacitances
PARASITIC CAP[A] CAP32 METAL3 METAL2
ATTRIBUTE CAP[A] 1 1
;
PARASITIC CAP[B] CAP21 METAL2 METAL1
ATTRIBUTE CAP[B] 1 1
;
; <functions omitted>
;
; METAL2/METAL3 parasitics
LEXTRACT CPA CAP32 NME3 BY CAP[A] CAPA32 &
EQUATION C = 1 * AREA + 1 * (PERI - OVPR) + 1 * OVPR
```

```
;
; METAL1/METAL2 parasitics
LEXTRACT CPA CAP21 NME1 BY CAP[B] CAPA21 &
EQUATION C = 1 * AREA + 1 * (PERI - OVPR) + 1 * OVPR
;
; build the netlist
LPESELECT[C] CAP[A] &
LPESELECT[C] CAP[B] OUTPUT CDL
*END
```

Correcting for Colinear Sidewall Capacitance

Sometimes, edges of two layers that make up a capacitor recognition layer are colinear. For example



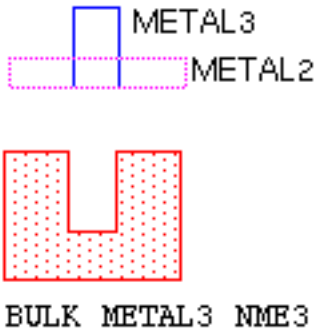
In this case, both sidewall-up and sidewall-down capacitances are calculated, which inaccurately represents the real capacitance value.

The two rules files that follow remove the double calculation for colinear edges:

- The first file splits parasitic capacitance into three subtypes for debugging purposes.
- The second file uses more efficient conjunctive commands.

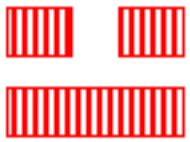
Sample Rules File for Debugging Colinear Capacitance

The following example illustrates how to extract and calculate the sidewall capacitances for the separate edges of the *METAL3*-to-*METAL2* capacitor layer.




```
NOT BULK METAL3 NME3
```

```
NOT BULK METAL2 NME2
```

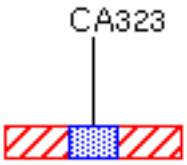


```
AND NME3 NME2 NME32
```

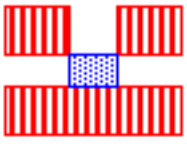
```
NOT NME3 NME32 NME3N2
```

```
NOT NME2 NME32 NME2N3
```

```
    AND METAL3 METAL2 CA32
    AND BULK CA32 CA323
    AND BULK CA32 CA3232
    AND BULK CA32 CA322
PARASITIC CAP[A] CA323  METAL3 METAL2 ...
PARASITIC CAP[B] CA3232 METAL3 METAL2 ...
PARASITIC CAP[C] CA322  METAL3 METAL2 ...
```




```
LEXTRACT CPA CA323 NME3N2 BY CAP[A] C323N2 &
EQUATION C = 1 * OVPR
```



```
LEXTRACT CPA CA3232 NME32 BY CAP[B] C3232 &
EQUATION C = 1 * OVPR
```

```
LEXTRACT CPA CA322 NME2N3 BY CAP[C] C322N3 &
EQUATION C = 1 * OVPR
```

The `NOT BULK METAL3` and `METAL2` operations produce “negatives” of the *METAL3* and *METAL2* layers. Other logical operations create areas for the `OVPR` function, which uses edges that touch or overlap.

The following example splits a parasitic capacitance into three subtypes for debugging purposes. A more efficient method for extracting this kind of capacitance when you don't need to debug is shown in the next section.

```
*DESCRIPTION
;
; <functions omitted>
;
; parameter set CPA specifies perimeter, overlap
; perimeter, and capacitance parameters
PARSET CPA OVPR C
;
*END
*INPUT-LAYER
;
; <functions omitted>
;
METAL3 = 1
METAL2 = 2
METAL1 = 3
SUBSTRATE = BULK 5
;
CONNECT-LAYER = BL2 BL1 BULK METAL1 METAL2 METAL3
*END
*OPERATION
;
; <functions omitted>
;
; create "negative" layers
NOT BULK METAL3 NME3
NOT BULK METAL2 NME2
;


; combine the "negative" layers
AND NME3 NME2 NME32
;
; create layer with non-colinear edges
NOT NME3 NME32 NME3N2
NOT NME2 NME32 NME2N3
;
; <functions omitted>
;
; create capacitor recognition layer
AND METAL3 METAL2 CA32; metal3-to-metal2 caps
;
; create capacitor recognition layers for each edge:
; each PARASITIC statement must have a unique
; capacitor recognition layer
;
AND BULK CA32 CA323
AND BULK CA32 CA3232
AND BULK CA32 CA322
;
; <functions omitted>
;
; specify parasitic capacitances
PARASITIC CAP[A] CA323 METAL3 METAL2
ATTRIBUTE CAP[A] 1 1
;
PARASITIC CAP[B] CA3232 METAL3 METAL2
```

```
ATTRIBUTE CAP[B] 1 1
;
PARASITIC CAP[C] CA322 METAL3 METAL2
ATTRIBUTE CAP[C] 1 1
;
; <functions omitted>
;
; extract the metal3 edges
LEXTRACT CPA CA323 NME3N2 BY CAP[A] C323N2 &
EQUATION C = 1 * OVPR
;
; extract the colinear edge
LEXTRACT CPA CA3232 NME32 BY CAP[B] C3232 &
EQUATION C = 1 * OVPR
;
; extract the metal2 edge
LEXTRACT CPA CA322 NME2N3 BY CAP[C] C322N3 &
EQUATION C = 1 * OVPR
;
; <functions omitted>
;
; build the netlist
LPESELECT[C] CAP[A] &
LPESELECT[C] CAP[B] &
LPESELECT[C] CAP[C] OUTPUT CDL
*END
```

Sample Rules File for Extracting Colinear Edge Capacitance

You can also extract colinear sidewall capacitance using conjunctive LEXTRACT/EQUATION functions. This sample rules file groups the three types of capacitance extracted in the previous section into a single subtype.

```
*DESCRIPTION
;
; <functions omitted>
;
; parameter set CPA specifies perimeter, overlap
; perimeter, and capacitance parameters
PARSET CPA OVPR P3 PO C
*END
*OPERATION
;
; <functions omitted>
;



; create "negative" layers
NOT BULK METAL3 NME3
NOT BULK METAL2 NME2
;
; combine the "negative" layers
AND NME3 NME2 NME32
;
; create layer with non-colinear edges
NOT NME3 NME32 NME3N2
NOT NME2 NME32 NME2N3
```

```
;
; <functions omitted>
;
; create capacitor recognition layers
AND METAL3 METAL2 CA32; metal3-to-metal2 caps
;
; specify parasitic capacitances
PARASITIC CAP[A] CA32 METAL3 METAL2
ATTRIBUTE CAP[A] 1 1
;
; conjunctive rules
LEXTRACT CPA CA32 NME3N2 BY CAP[A] C32 & ; metal3 edges
EQUATION P3 = 1 * OVPR &; save to P3
LEXTRACT CPA CA32 NME32 BY CAP[A] &; colinear edges
EQUATION PO = 1 * OVPR &; save to PO
;
; <functions omitted>
;
LEXTRACT CPA CA32 NME2N3 BY CAP[A] &; metal2 edge
EQUATION C = 1 * OVPR + P3 + P
;
; <functions omitted>
;
; build the netlist
LPESELECT[C] CAP[A] OUTPUT CDL
*END
```

Extracting Single-Layer Fringe Capacitance

You can extract the fringe capacitance between edges on a single layer as shown in the sample rules file that follows. Dracula checks edges within the distance you specify, and ignores the others.

```
*DESCRIPTION
;
; <functions omitted>
UNIT = CAPACITANCE,PF
;
*END
*INPUT-LAYER
;
; <functions omitted>
METAL = 6
;
; <functions omitted>
*END
*OPERATION
;
; <functions omitted>
PARASITIC CAP[MF] METAL METAL METAL ; fringe capacitors on
ATTRIBUTE CAP[MF] 10 0.00005; the METAL layer
;
; <functions omitted>
LPESELECT[S] CAP GT 0.0 OUTPUT SPICE
*END
```

The values that you supply with the ATTRIBUTE CAP statement vary depending upon the type of capacitance you are extracting. The values for single-layer fringe capacitance are

```
ATTRIBUTE CAP separation capacitance
```

separation

The maximum separation between edges. In the previous rules file, capacitance is calculated for metal edges within 10 units or less of each other.

capacitance

The capacitance multiplier. The capacitance value is 0.00005 picofarads per perimeter unit in the previous rules file. The capacitance unit is set to PF with the UNIT function in the Description block.

Extracting Two-Layer Fringe Capacitance

This section shows how to extract the fringe capacitance between edges on two different layers. This rules file uses the FRINGE CAP function, which extracts capacitance between the edges of different nodes. Capacitance is not extracted for edges of the same electrical node.

```
*DESCRIPTION
;
; <functions omitted>
;
UNIT = CAPACITANCE,PF
*END
*INPUT-LAYER
;
; <functions omitted>
;
METAL3 = 1
METAL2 = 2
METAL1 = 3
*END
*OPERATION
;
; <functions omitted>
;
FRINGE CAP[A] METAL1 METAL2
ATTRIBUTE CAP[A] 0.9 0.9
;
FRINGE CAP[B] METAL1 METAL3
ATTRIBUTE CAP[B] 0.8 0.8
;
FRINGE CAP[C] METAL2 METAL3
ATTRIBUTE CAP[C] 0.9 0.9
;
; <functions omitted>
;
LPESELECT[A] CAP[A] &
LPESELECT[A] CAP[B] &
```

```
LPESELECT[A] CAP[C] OUTPUT CDL
*END
```

The A option for the LPESELECT function includes power and ground nodes in the extraction. Because the C option, which outputs cross-coupled rather than lump-sum capacitances, conflicts with the A option, you can't use it. Instead, use the T option, which includes the power and ground signals in the coupling report.

Note: The T option is not supported for PRE.

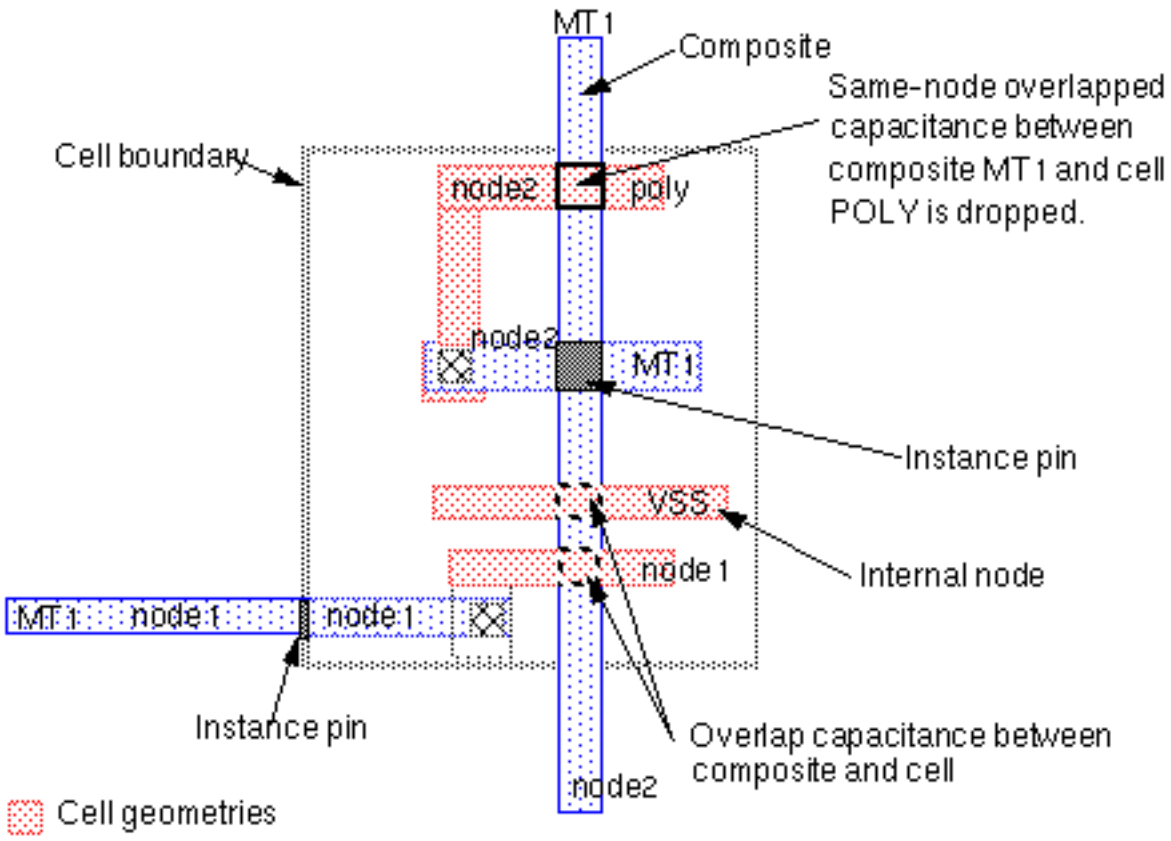
Extracting Over-the-Cell Routing Wire Overlap Capacitance with Cell Geometries in HLPE

Dracula HLPE/HPRE does not extract composite-to-cell overlapped parasitic capacitors in composite mode by default. To extract these overlap capacitances, use the EXPLODE function.

This function flattens the cell geometries to the composite plane. It assigns composite node numbers to nodes connected to the composite plane, as shown for node1 and node2 in the figure. External nodes are cell nodes that connect to the composite plane by instance pins.

Dracula connects signals that are completely internal to the Hcell to a signal that you specify in the EXPLODE function. Internal signals are connected to VSS in this example:

```
EXPLODE POLY GPOLY VSS
```



The sample rules file that follows shows two ways of using the `EXPLODE` function:

- Without the `A` option

  The output layer contains only the flattened cell data.

- With the `A` option

  The output layer contains the merged composite and flattened cell geometries.

```
*DESCRIPTION
CHECK-MODE = COMPOSITE
;
; <functions omitted>
;
CONNECT-LAYER = POLY MT1 MT2
;
; <functions omitted>
;
CONNECT MT2 MT1  BY VIA
CONNECT MT1 POLY BY CONT
;
```

```
; <functions omitted>
;
; EXPLODE{[A]} input-layer output-layer labelname
; Without the A option specified in the EXPLODE function,
; the output layer contains only the flattened cell data.
; When the A option is used, the output
; layer contains the merged data of composite and
; flattened cell geometries.
;
; Using the [A] option
EXPLODE[A] POLY GPOLY VSS
EXPLODE[A] MT1  GMET1 VSS
AND  MT2   GMT1  M2AM1    composite MT2 to all MT1

AND  GMT1 GPOLY AM1APL   ; all MT1 to all POLY,
                                ; including
                                ; cell-to-cell overlap
                                ; capacitance
;
; Without the [A] option
EXPLODE MT1 XMET1 VSS
AND  MT2 MT1   M2M1             ; comp MT2 to comp MT1
AND  MT2 XMET1 M2CM1            ; comp MT2 to cell MT1
EXPLODE POLY XPOLY VSS
AND MT1 POLY  M1PL              ; comp MT1 to comp POLY
AND  MT1 XPOLY M1CPL            ; comp MT1 to cell POLY
AND XMET1 XPOLY CM1CPL          ; cell MT1 to cell POLY
;
; <functions omitted>
;
PARASITIC CAP[A] M2AM1 MT2 GMT1
ATTRIBUTE CAP[A] 1 1
PARASITIC CAP[B] AM1APL GMT1 GPOLY
```

Note: When you use the `EXPLODE` function in the 4.1hot version, add `EXCEPTION-ON = [EXPLODE]` as the first line in the Description block.

Extracting Parasitic Resistance

The basic types of parasitic resistors that you might want to extract are

- Simple metal and poly resistors
- Two-layer metal parasitics
- Contact resistors
- Diffusion resistors

You do not need to extract corner and junction resistors explicitly, because the `CUT-TERM` function automatically extracts them. For more information about corner and junction resistors, see “Calculating Junction Resistance” in the “Extracting Parasitic Resistance (PRE)” chapter of the *Dracula Reference.*

You might also need to include rules in your file that

- Correct sidewall capacitance to remove resistor edges
- Extract fringe capacitance in PRE
- Control how contacts are cut

To extract resistance, do the following in your rules file:

1. Create the resistor body and terminal layers.
2. Specify equations for calculating parasitic values.

This section might not cover all the types of resistance that you want to extract. However, you can use the examples in this section as general guidelines to develop similar methods for extracting parasitic resistance from your layout.

Preparing to Extract Resistance

To prepare for coding resistance extraction in your Dracula rules file, do the following:

- Extract and verify capacitance
- Specify a pad layer
- Be familiar with the Dracula functions for resistance extraction

These preparations are described in the sections that follow.

Extracting Capacitance

Because resistors change a single node into multiple nodes, always extract capacitors before you extract resistors. This process lets you verify that you have extracted capacitance correctly before you add resistor extraction to the rules file.

After you add resistance, validate that the capacitance is being extracted correctly by comparing the capacitance reported by the capacitance-only run with the results from the RC extract run.

Specifying a Pad Layer

When resistor extraction divides nodes into subnodes, Dracula uses the pad layer to determine which subnode represents a cell I/O. If you don’t specify a pad layer, Dracula might choose a node that might not be used as an I/O when the cell is connected into a circuit.

Depending on the level at which you are running Dracula, you can specify the pad layer as follows:

- Top-level cell — the pad layer you digitized in the layout
- Lower cells — the pins you specified to a place-and-route tool when you created the layout

Internal nodes do not need a pad layer.

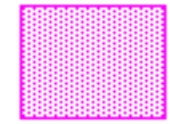
If you define a “pad layer” in a lower level cell, it is ignored when you run Dracula on the upper level cell. Dracula also ignores pins on nodes that are inside a block created by a place-and-route tool.

If you cannot get the pad layer directly from the layout, here’s how to generate a pad layer:

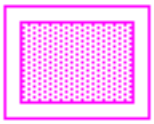
```
*INPUT-LAYER
;
; <functions omitted>
;
METAL = 6
PAD-LAYER = PAD
SUBSTRATE = BULK 8
*END
*OPERATION
;
; <functions omitted>
;
; Create pad layer
SIZE BULK BY -2 NOPAD
NOT METAL NOPAD PAD
;
; <functions omitted>
;
*END
```

Note: The `PAD-LAYER` function terminates resistances at geometries on the specified layer.

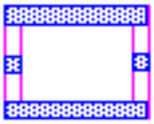
The subnodes that represent the I/O for a sample layout are created with the NOT function as shown in the following figure.



SUBSTRATE = BULK



SIZE BULK BY -2 NOPAD



NOT METAL NOPAD PAD

Caution

This method does not work for a composite run.

For composite mode, you must first flatten the bulk layer to create a pad layer similar to the one in flat mode:

```
FLATTEN BULK ABULK
FLATTEN METAL AMETAL
SIZE ABULK BY -2 NOPAD
NOT AMETAL NOPAD APAD
HIERARCHEN APAD PAD
```

Using the Extraction Functions

The table in this section lists

- The steps for extracting resistors from your layout
- Which functions to use for each step

- Where to put the functions in your Dracula rules file

Resistance Extraction Flow

Step	Function	Description	Rules File Block
Define the resistance unit	`UNIT`	Associates natural units with extracted parameters.	`*DESCRIPTION`
Define the subnode delimiter	`SUBNODE-DELIM`	Specifies the delimiter in subnode names (optional).	
Create the resistor	`CUT-TERM`	Generates resistor body and resistor terminal layers from a conductor layer and a contact layer.	`*OPERATION`
	`STAMP`, `AND`, or `NOT`	Passes node information to resistor terminals so that the network can be reconnected properly.	
Reconnect the network	`RCONNECT-LAYER`	Specifies the mask order in which to reconnect the layout circuit after the conductor nodes have been cut into parasitic resistors.	`*INPUT-LAYER`
	`RCONNECT`	Reconnects layers after conductor nodes are cut into parasitic resistors.	
	`PAD-LAYER`	Terminates resistances at the specified pad layer; not needed if you are at the top level of the design.	
Specify parasitic devices and resistance values	`PARASITIC RES` and `ATTRIBUTE RES`	Specifies the parasitic resistor body and terminal layers, and supplies the sheet resistance in ohms per square unit.	`*OPERATION`
Generate output	`LEXTRACT`	Extracts parameters from the resistor layers (optional).	

Resistance Extraction Flow, *continued*

Step	Function	Description	Rules File Block
Generate output (cont'd)	EQUATION	Used with LEXTRACT to specify the equation used to compute extracted resistance. Uses the device parameters specified in the PARSET function. The LEXTRACT/ EQUATION combination is called Flex LPE (optional).	*OPERATION
	LPESELECT	Writes the parasitic resistance to an output SPICE file.	
	SPFSELECT	Writes the parasitic resistance to an output DSPF file (for HPRE jobs only).	
	RSPFSELECT	Writes the parasitic resistance to an output RSPF file (for HPRE jobs only).	

Creating Resistor Recognition Layers

You can create the recognition layer for a parasitic resistor using one of these functions:

- The CUT-TERM function
- The CUT function
- Logical and spacing functions

Using CUT-TERM is the easiest method because you only need one function to create parasitic resistor layers. The *Dracula Reference* has a complete description of the CUT-TERM function in the Operation Block Commands chapter. A number of sample rules files in this chapter use the CUT-TERM function.

The CUT function gives you more control over resistor creation than the CUT-TERM function, but requires you to write about 10 additional lines of Dracula code.

Logical and spacing functions can't duplicate all the processing that CUT-TERM and CUT do. You use logical and spacing functions to cut the resistor recognition layer out of irregular shapes, such as source/drains. For details, see the example in "Extracting Diffusion Resistors" on page 111.

Extracting Simple Metal and Poly Resistors

This section describes how to extract metal and poly resistors from a silicon-gate CMOS layout. The `CUT-TERM` function is used to create resistor body and terminal layers.

```
;
; <functions omitted>
;
*INPUT-LAYER
PWELL = pwell
ACTIVE = active
POLY = poly
PDIFF = pdiff
CONT = contact
MET = metal
VAPOX = vapox
;
; <functions omitted>
;
CONNECT-LAYER = NWELL PWELL PSD NSD POLY MET
;
; <functions omitted>
;
; reconnect the network
RCONNECT-LAYER = NWELL PWELL PSD NSD PLYTRM METTRM
*END
*OPERATION
;
; <functions omitted>
;
AND ACTIVE PDIFF PPLUS
NOT ACTIVE PPLUS NPLUS
AND POLY PPLUS PGATE
AND POLY NPLUS NGATE
NOT NPLUS NGATE NSD
NOT PPLUS PGATE PSD
AND PSD PWELL PCONT
AND NSD NWELL NCONT
;
; <functions omitted>
;
; <insert layer interconnections with CONNECT here>
;
; <include ELEMENT and LVSCHK/LPECHK functions here>
;
; create the resistors
CUT-TERM MET CONT METRES METTRM;create metal resistors
STAMP METTRM BY MET;transfer node info
;
OR NGATE PGATE GATE;combine all gate areas
CUT-TERM POLY CONT PLYRES PLYTRM GATE ; create poly res.
STAMP PLYTRM BY POLY;transfer node info
;
; <functions omitted>
;
; re-connect the network
RCONNECT METTRM PLYTRM BY CONT
RCONNECT METTRM PSD BY CONT
RCONNECT METTRM NSD BY CONT
RCONNECT PSD PWELL BY PCONT
```

```
RCONNECT NSD NWELL BY NCONT
;
; <functions omitted>
;
; define parasitic resistor devices & values
PARASITIC RES[ME] METRES METTRM ; metal resistor
ATTRIBUTE RES[ME] 0.06;sheet resistance, ohms/sq
;
PARASITIC RES[PL] PLYRES PLYTRM ; poly resistor
ATTRIBUTE RES[PL] 25.0;sheet resistance, ohms/sq
;
; <functions omitted>
;
; generate output for SPICE file
LPESELECT[S] RES GE 10 OUTPUT SPICE
;
; <functions omitted>
;
; generate output for DSPF file
SPFSELECT[S] RES GE10 OUTPUT SPF
*END
```

Extracting Two-Layer Metal Parasitics

When you extract resistance and capacitance from a design with two metal layers, you must `OR` the contact and via layers into a single layer. This layer is used as the contact layer for the `CUT-TERM` function on the first metal layer. The following rules file, which extracts both capacitance and resistance, uses this technique for a double-metal CMOS process:

```
;
; <functions omitted>
;
*INPUT-LAYER
NDIFF=1
NWELL=2
POLY=4
CONT=5
MET1=6
PDIFF=7
VAPOX=8
VIA=9
MET2=10
SUBSTRATE = BULK 64
CONNECT-LAY= PWELL NWELL PSD NSD POLY MET1 MET2
RCONNECT-LAY= PWELL NWELL PSD NSD POLY MTRM M2TRM
PAD-LAYER = VAPOX  ; pad at passivation layer
*END
;
*OPERATION
;
; <functions omitted>
;
NOT BULK NWELL PWELL
NOT PDIFF POLY PACTIVE
NOT NDIFF POLY NACTIVE
AND POLY NDIFF NGATE
AND POLY PDIFF PGATE
```

```
OR NGATE PGATE GATE
AND NACTIVE NWELL NWCONT
NOT NDIFF NGATE NSD
NOT PDIFF PGATE PSD
;
; <functions omitted>
;
LVSCHK
;
;************************************************
; Here begins the second pass - which performs    *
; the parasitic resistor extraction               *
;************************************************
;
;Cutting resistors on MET1 layer
;OR all of the contact layers together before cut-term
;
OR VIA CONT MCON
CUT-TERM  MET1  MCON  MRES  MTRM
STAMP MTRM BY MET1
;
;Cutting resistors on MET2 layer
;
CUT-TERM  MET2  VIA  M2RES  M2TRM
STAMP M2TRM BY MET2
;
RCONNECT M2TRM MTRM BY VIA
RCONNECT NSD NWELL BY NWCONT
RCONNECT PSD PWELL BY PWCONT
RCONNECT MTRM POLY BY CONT
RCONNECT MTRM NSD BY CONT
RCONNECT MTRM PSD BY CONT
;
AND NSD PWELL NDIO
NOT PSD PWELL PDIO
LINK BULK TO VSS
;
; *** MET1 ***
NOT MTRM POLY MNTRP ; m1 resistor terminal not over poly
NOT MRES POLY MNREP       ; m1 resistor not over poly
AND MTRM POLY MTPLY       ; m1 resistor terminal to poly
AND MRES POLY MRPLY       ; m1 resistor to poly
AND MNTRP DIFF MNTD       ; m1 resistor terminal to diff cap
AND MNREP DIFF MNRD       ; m1 resistor to diff cap
NOT MNTRP MNTD MNTND      ; m1 resistor terminal not over
                          ; poly,diff
NOT MNREP MNRD MNRND ; m1 resistor not over poly,diff
AND MNTND BULK MTF   ; m1 resistor terminal to field cap
AND MNRND BULK MRF   ; m1 resistor to field cap
;
; *** MET2 ***
NOT M2TRM MET1 M2NTM ; m2 resistor terminal not over m1
NOT M2RES MET1 M2NRM ; m2 resistor not over m1
AND M2TRM MTRM  M2TT ; m2 resistor terminal to m1 RT cap
AND M2RES MTRM  M2RT ; m2 resistor to m1 RT cap
AND M2TRM MRES  M2TR ; m2 resistor terminal to m1
                     ; resistor cap
AND M2RES MRES  M2RR ; m2 resistor to m1 resistor cap
AND M2NTM POLY  M2TPLY ; m2 resistor terminal to poly cap
AND M2NRM POLY  M2RPLY ; m2 resistor to poly cap
NOT M2NTM POLY M2TMP   ; m2 resistor terminal not over
```

```
                        ; metal,poly
NOT M2NRM POLY M2RMP    ; m2 resistor not over metal,poly
AND M2TMP DIFF M2TD     ; m2 resistor terminal to diff cap
AND M2RMP DIFF M2RD     ; m2 resistor to diff cap
NOT M2TMP M2TD M2NTD    ; m2 resistor terminal not over
                        ; metal, poly, diff
NOT M2RMP M2RD M2NRD    ; m2 resistor not over
                        ; metal, poly, diff
AND M2NTD BULK M2TF     ; m2 resistor terminal to field cap
AND M2NRD BULK M2RF     ; m2 resistor to field cap
;
; *** POLY ***
NOT  POLY  GATE  PLY1
AND  PLY1  BULK  PTF ; poly (exclude GATE) to field cap.
;
; <functions omitted>
;
PARASITIC RES[R1] MRES MTRM
ATTRIBUTE RES[R1] 94
;
PARASITIC RES[R2] M2RES M2TRM
ATTRIBUTE RES[R2] 60
;
; *** Fringe capacitance ***
FRINGE[R] CAP[FA] M2RES M2TRM  ; m2 fringe capacitance
ATTRIBUTE CAP[FA] 10    .0093
;
FRINGE[R] CAP[FB] MTRM  MRES ;MRES fringe capacitance
ATTRIBUTE CAP[FB] 10    .0079
;
; *** Overlap capacitance ***
PARASITIC CAP[P1] MTPLY MTRM POLY ;M1 terminal to POLY cap
ATTRIBUTE CAP[P1] .00596  0.0123
;
PARASITIC CAP[P2] MRPLY MRES POLY ;M1 resistor to POLY cap
ATTRIBUTE CAP[P2] .00596   0.0123
;
PARASITIC CAP[D1] MNTD MTRM DIFF ;M1 terminal to DIFF cap
ATTRIBUTE CAP[D1] .00596  0.0126
;
PARASITIC CAP[D2] MNRD MRES DIFF ;M1 resistor to DIFF cap
ATTRIBUTE CAP[D2] .00596  0.0126
;
PARASITIC CAP[F1] MTF MTRM BULK ;M1 terminal to FIELD cap
ATTRIBUTE CAP[F1] .0033  0.0098
;
PARASITIC CAP[F2] MRF MRES BULK ;M1 resistor to FIELD cap
ATTRIBUTE CAP[F2] .0033  0.0098
;
PARASITIC CAP[M1] M2TT M2TRM MTRM   ;M2 terminal to
ATTRIBUTE CAP[M1] .0084  0.0101     ;M1 terminal cap
;


PARASITIC CAP[M2] M2RT M2RES MTRM    ;M2 resistor to
ATTRIBUTE CAP[M2] .0084  0.0101      ;M1 terminal cap
;
PARASITIC CAP[M3] M2TR M2TRM MRES    ;M2 terminal to
ATTRIBUTE CAP[M3] .0084  0.0101      ;M1 resistor cap
;
PARASITIC CAP[M4] M2RR M2RES MRES    ;M2 resistor to
```

```
ATTRIBUTE CAP[M4] .0084  0.0101      ;M1 resistor cap
;
PARASITIC CAP[P3] M2TPLY M2TRM POLY   ;M2 terminal to
ATTRIBUTE CAP[P3] .0037  0.0067       ;POLY cap
;
PARASITIC CAP[P4] M2RPLY M2RES POLY   ;M2 resistor to
ATTRIBUTE CAP[P4] .0037  0.0067       ;POLY cap
;
PARASITIC CAP[D3] M2TD M2TRM DIFF ;M2 terminal to DIFF cap
ATTRIBUTE CAP[D3] .0033  0.0062
;
PARASITIC CAP[D4] M2RD M2RES DIFF ;M2 resistor to DIFF cap
ATTRIBUTE CAP[D4] .0033  0.0062
;
PARASITIC CAP[F3] M2TF M2TRM BULK ;M2 terminal to
ATTRIBUTE CAP[F3] .0053  0.0026   ;field cap
;
PARASITIC CAP[F4] M2RF M2RES BULK ;M2 resistor to
ATTRIBUTE CAP[F4] .0053  0.0026   ;field cap
;
PARASITIC CAP[P5] PTF POLY BULK   ;POLY to field cap
ATTRIBUTE CAP[P5] .0065  0.0076
;
; <functions omitted>
;
LPESELECT[S] CAP GT 0.0 &
LPESELECT[S] RES GT 0.0 OUTPUT SPICE
;
*END
```

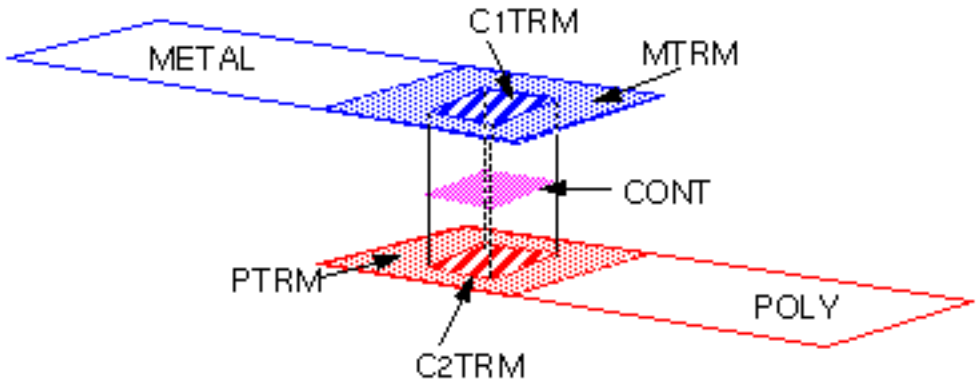
Extracting Contact Resistors

The partial rules file in this section extracts simple parasitic metal and poly resistors, as well as metal/poly contact resistors.

The current flow of the parasitic resistors on the interconnect wire is horizontal, while the current flow of the contact resistors is vertical. To form contact resistors, you create two `RCONNECT-LAYER` layers. These layers are the terminal layers of the contact resistors on the upper and lower layers to which the contact connects.

To form contact resistors, break the connection between the two `RCONNECT-LAYER` layers and define the contact between them as the contact resistor body layer. The two `RCONNECT-LAYER` layers become the terminal layers on the upper and lower sides of the contact resistors.

The following figure demonstrates this concept, where *C1TRM* and *C2TRM* are the `RCONNECT-LAYER` layers:



This rules file combines contacts to metal and poly resistors on one contact resistor recognition layer.

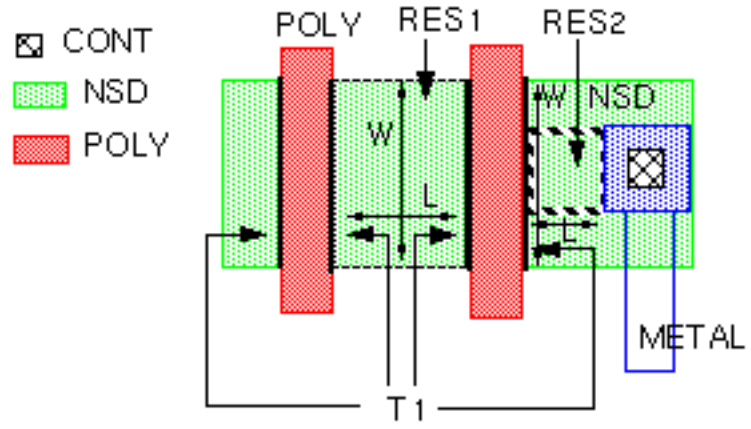
```
*DESCRIPTION
;
; <functions omitted>
;
PARSET = CTR1 AREA R
*END
*INPUT-LAYER
PWELL     = 1
ACTIVE    = 2
POLY      = 3
PIMP      = 4
CONT      = 5
METAL     = 6
SUBSTRATE = BULK 99
;
CONNECT-LAYER  = NSUB PWELL PSD NSD POLY METAL
RCONNECT-LAYER = NSUB PWELL PSD NSD PTRM MTRM
PAD-LAYER= VAPOX
*END
*OPERATION
;
; <functions omitted>
;
NOT BULK PWELL NSUB
AND ACTIVE PIMP PPLUS
NOT ACTIVE PPLUS NPLUS
AND POLY PPLUS PGATE
AND POLY NPLUS NGATE
NOT NPLUS NGATE NSD
NOT PPLUS PGATE PSD
AND PSD PWELL PTAP
AND NSD NSUB NTAP
```

```
;
; connect the circuit
;
CONNECT METAL POLY BY CONT
CONNECT METAL PSD BY CONT
CONNECT METAL NSD BY CONT
CONNECT PSD PWELL BY PTAP
CONNECT NSD NSUB BY NTAP
;
; <functions omitted>
;
; cut resistors on poly layer
;
OR NGATE PGATE GATE; all gates on interconnect layer
CUT-TERM POLY CONT PRES PTRM GATE
STAMP PTRM BY POLY
;
; cut resistors on metal layer
;
CUT-TERM METAL CONT MRES MTRM
STAMP MTRM BY METAL
;
; form contact resistor layer
;
AND MTRM CONT C1TRM   ; create contact resistor
                      ; terminal layer 1
AND PTRM CONT C2TRM   ; create contact resistor
                      ; terminal layer 2
AND PTRM CONT P2CONT  ; create dummy contact layer
AND C1TRM C2TRM MPRES ; create contact resistor
                      ; recognition layer
;
; reconnect the circuit
;
RCONNECT MTRM C1TRM BY CONT; MTRM --> C1TRM
RCONNECT MTRM PSD BY CONT
RCONNECT MTRM NSD BY CONT
RCONNECT C2TRM PTRM BY P2CONT ; C2TRM --> PTRM
RCONNECT PSD PWELL BY PTAP
RCONNECT NSD NSUB BY NTAP
;
; <functions omitted>
;
; define parasitic capacitances
;
PARASITIC RES[P] PRES PTRM; poly interconnect resistor
ATTRIBUTE RES[P] 0.01; Kohms per square
;
PARASITIC RES[M] MRES MTRM; metal interconnect res.
ATTRIBUTE RES[M] 0.01; Kohms per square
;
PARASITIC RES[C] MPRES C1TRM C2TRM; metal/poly
ATTRIBUTE RES[C] 0.01             ; contact resistors
;
; Using Flexible LPE to calculate the contact resistor
LEXTRACT CTR1 MPRES BY RES[C] CTRES &
EQUATION R = 0.1 / AREA; contact resistance equation
;
; output results
;
LPESELECT RES GT 0.0 OUTPUT SPICE
```

```
;
*END
```

Extracting Diffusion Resistors

The geometric shape of diffusion layers is different from that of interconnect layers. The following example uses DRC and logical operations to form the terminal layer of the diffusion resistors.



Note: Form *RES2* using contact, not metal.

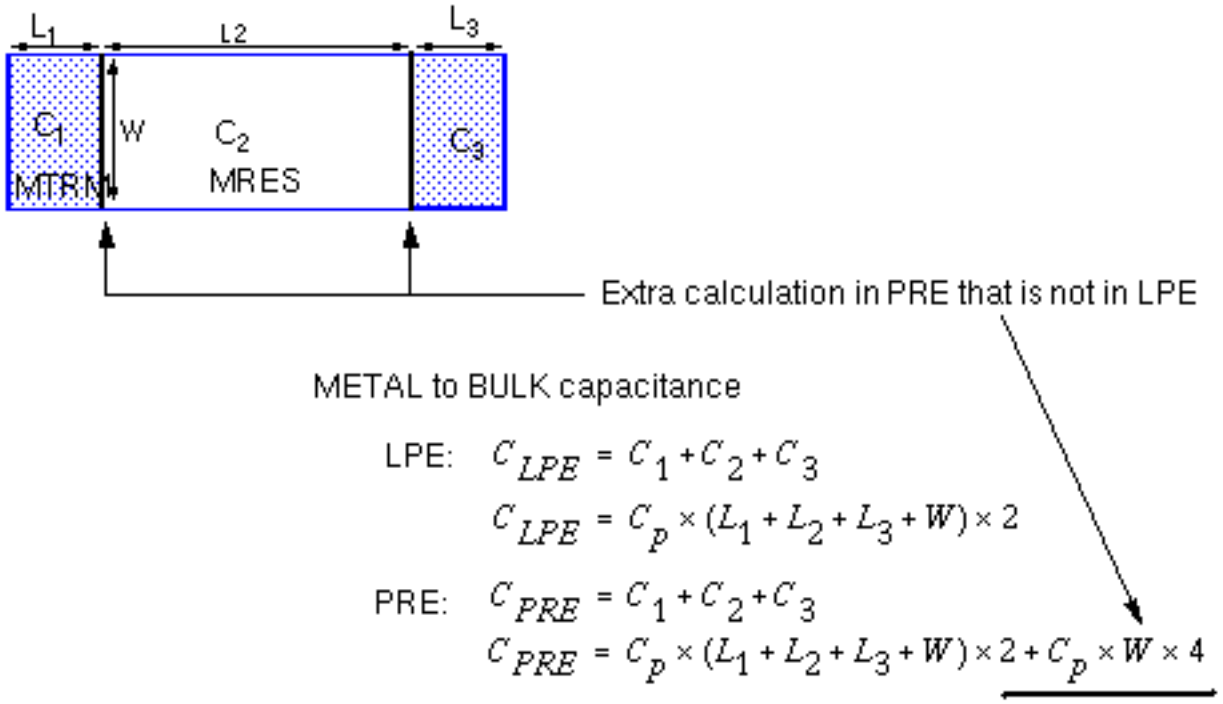
```
*DESCRIPTION
PARSET PR1 AREA W1 W2 L W R
PARSET PR2 OVPR L W R
;
; <functions omitted>
;
CONNECT-LAYER = PSD NSD POLY METAL
RCONNECT-LAYER= PSDT NSDT POLY METAL
;
*END
*OPERATION
;
; <functions omitted>
;
AND POLY NDIF NGAT
NOT NDIF NGAT NSD
;
; <functions omitted>
;
CONNECT METAL NSDBY CONT
;
; <functions omitted>
;
ELEMENT MOS[N]NGAT POLY NSD
;
; <functions omitted>
```

```
;
AND METAL CONT MCON
AND MCON  NSD  SCON
SIZE    NGATE BY 0.001 SNGAT
AND     NSD   SNGAT T1
NOT     NSD   T1    SUB1
SELECT  SUB1  TOUCH [2:2] T1 RES1
NOT     SUB1  RES1  SUB2
SIZE    SCON  BY 0.8 BCON
EXT[RC] BCON  NGATE LT 10 R2
AND     SUB2  R2    RES2
AND     NSD   BCON  T2
OR      T1    T2    NSDT
STAMP   NSDT  BY    NSD
;
; <functions omitted>
;
RCONNECT METAL NSDT BY CONT
;
; <functions omitted>
;
ELEMENT MOS[N] NGAT POLY NSDT
;
; <functions omitted>
;
PARASITIC RES[R1] RES1 NSDT
ATTRIBUTE RES[R1]1
PARASITIC RES[R2] RES2 NSDT
ATTRIBUTE RES[R2]1
;
LEXTRACT PR2 RES2 T1 BY RES[R2] NRES2 &
EQUATION W = OVPR
LEXTRACT PRES RES2 T2 BY RES[R2] &
EQUATION L = (PERI - W -OVPR)/2
EQUATION R = 1 * L/W
;
LEXTRACT PR1 RES1 NSDT BY RES[R1] NRES1 &
EQUATION W = (W1+W2)/2 &
EQUATION L = AREA/W &
EQUATION R = 1 * L/W /.5
```

Correcting Sidewall Capacitance

When you specify the perimeter attribute to extract and calculate the sidewall capacitance of overlapped capacitors in PRE, the resulting capacitance value is inaccurate when compared

to the same value extracted in LPE. The inaccuracy occurs because the calculation includes the edges formed when the layer was cut into resistors.



The following is an excerpt from a Dracula rules file that corrects that problem.

```
*DESCRIPTION
;
; <functions omitted>
;
PARSET CPA PERI OVPR
;OPERATION
;
; <functions omitted>
;



; cut resistors on metal layer
;
CUT-TERM METAL CONT MRES MTRM
STAMP MTRM BY METAL
RCONNECT MTRM POLY BY CONT
;
; <functions omitted>
;
; define parasitic capacitance
AND MTRM BULK MTCAP
AND MRES BULK MRCAP
;
PARASITIC CAP[A] MTCAP MTRM BULK
```

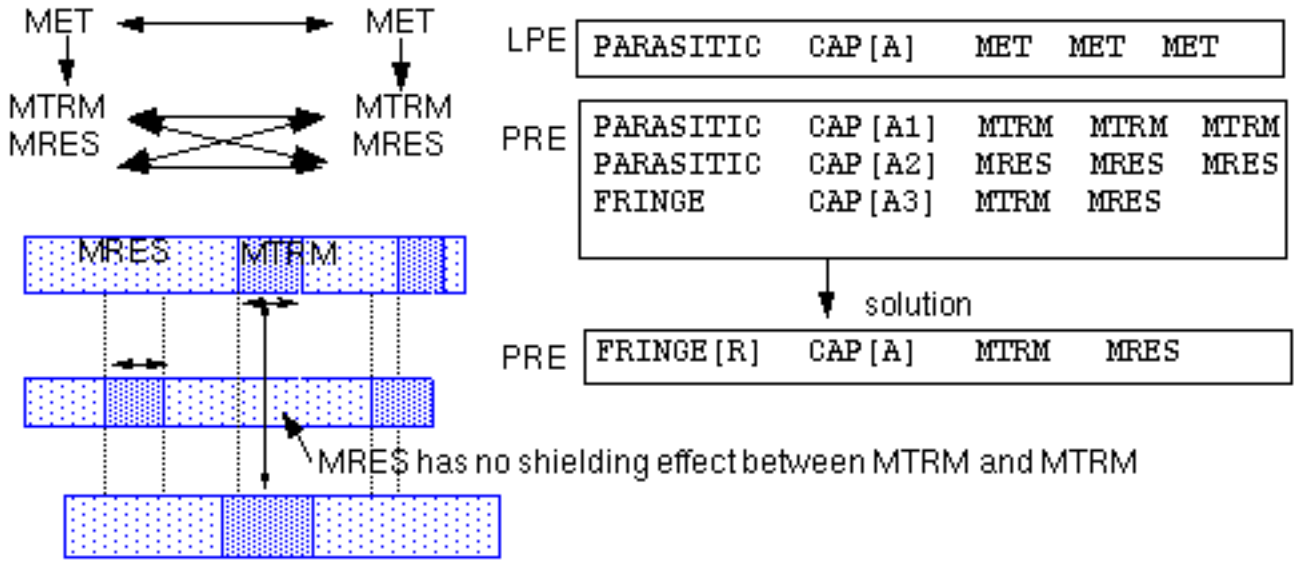
```
ATTRIBUTE CAP[A] 0 1
PARASITIC CAP[B] MRCAP MRES BULK
ATTRIBUTE CAP[B] 0 1
;
LEXTRACT CPA MTCAP MRCAP BY CAP[A] CAP1&
EQUATION C = 1 * (PERI-OVPR)
;
LEXTRACT CPA MRCAP MTCAP BY CAP[B] CAP2&
EQUATION C = 1 * (PERI - OVPR)
;
; <functions omitted>
```

Extracting Fringe Capacitance in PRE

When you extract the fringe capacitance of interconnect wires in PRE, you generally use two `PARASITIC CAP` functions and one `FRINGE CAP` function:

- Parasitic resistor terminal to parasitic resistor terminal (*MTRM* to *MTRM*)
- Parasitic resistor body to parasitic resistor body (*MRES* to *MRES*)
- Parasitic resistor terminal to parasitic resistor body (*MTRM* to *MRES*)

The following example shows how you can use a single `FRINGE[R] CAP` function to extract the fringe capacitance of the resistor body and terminal layers resulting from `CUT-TERM`. When you use the `[R]` option of the `FRINGE` function in PRE, you can get the same capacitance value that you get using LPE.




```
;
; <functions omitted>
```

```
;
CONNECT-LAYER = POLY METAL
RCONNECT-LAYER = POLY MTRM
;
; <functions omitted>
;
CONNECT METAL POLY BY CONT
;
; <functions omitted>
;
CUT-TERM METAL CONT MRES MTRM
;
; <functions omitted>
;
RCONNECT MTRM POLY BY CONT
;
; <functions omitted>
;
FRINGE[R] CAP[M] MRES MTRM
ATTRIBUTE CAP[M] 5.0 0.0001
;
; <functions omitted>
```

Controlling How Contacts Are Cut

The CUT-TERM function enlarges contacts when cutting resistors, so not all resistors are the same size. To control how contacts are cut, use the CUT function rather than CUT-TERM.

Note: If you need to control the maximum number of squares to break resistors into, use the MAXNS option of the CUT-TERM function. If you use CUT, there is no way to control how the resistors are divided.

The syntax of the CUT function is

```
CUT [STRIP] cutLayer stripLayer outputLayer
```

cutLayer

The layer to cut.

stripLayer

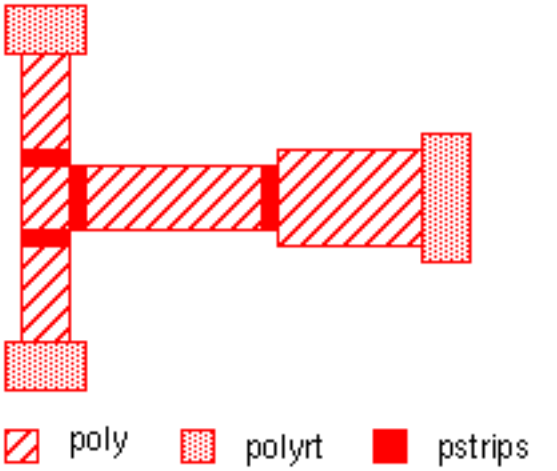
Used with the STRIP option. This is the parasitic resistor terminal layer that guides the strip cuts in a direction perpendicular to the flow of current.

outputLayer

Contains the results of the CUT function.

Here is an example of the CUT command:

```
CUT STRIP POLY PSTRIPS POLYRT
```



The rules file excerpt that follows uses CUT instead of CUT-TERM.

```
;
; <functions omitted>
;
*INPUT-LAYER
CONT     = 1
VIA      = 2
DIFF     = 4
METAL1   = 5
METAL2   = 6
RES      = 7
;
; <functions omitted>
;
SUBSTRATE = BULK 8
TEMPORARY-LAYER = SUB1 SUB2 SUB3 SUB4
*END
*OPERATION
;
; <functions omitted>
;
; build "negative" layer for filtering out edges
; formed by resistor cuts
;
NOT BULK METAL2 NOM2
;
; <functions omitted>
;
; combine contact and resistor device layer to define
; where not to build resistors
;
```

```
OR VIA RES ALLCON
;
; create resistor recognition layers for RES[A}
;
NOT    METAL2 ALLCON SUB1
WIDTH[CR] SUB1 LT 2 SUB2
NOT    SUB1   SUB2   SUB3
OR     ALLCON SUB3   SUB4
CUT    STRIP  SUB2   SUB4 SUB1
OR     SUB4   SUB1   SUB3
NOT    SUB2   SUB1   SUB4
SELECT SUB4   TOUCH[2:2] SUB3 M2RES
NOT    SUB4   M2RES  SUB1
OR     SUB1   SUB3   OUTRM
;
; <functions omitted>
;
; transfer nodal information to resistor terminals
;
AND METAL2 OUTRM M2TRM
;
; reconnect with nodes divided by resistors
RCONNECT M2TRM DIFF BY VIA
;
; <functions omitted>
;
; extract parasitic devices
;
PARASITIC RES[A] M2RES M2TRM
ATTRIBUTE RES[A] 1
;
LPESELECT[Y] RES[A] &
.....................OUTPUT CDL
*END
```

Extracting 2.5D MB Parasitics

Capacitors close together affect each others' capacitance, a fact that is reflected by the term "2.5 dimensional multibody" (2.5D MB) capacitance. Standard capacitance extraction does not account for this interaction between capacitors sharing the same plate. When you extract only simple capacitance, the effects of other capacitors sharing the same plate are ignored.

As the spacing between geometries shrinks, the reaction between sidewall and fringe capacitors becomes an important consideration. There are six possible combinations of two different capacitors affecting the same point on one plate. You need to consider not only one point on a plate but the area of the plate, so the interrelationship grows exponentially.

Dracula 2.5D MB Capabilities

You can use the ATTRIBUTE CAP function in your rules file to have Dracula extract simple RC and 2.5D MB parasitics. For a description of ATTRIBUTE CAP, see the ATTRIBUTE-CAP function in the “Operation Block Commands” chapter of the *Dracula Reference* manual.

Note: Where you need more accuracy, you can also use a tool that does linear equation analysis. If you want to use such a hybrid approach, contact your local Field Applications Engineer for assistance.

The following sections describe how Dracula 2.5D MB capabilities can extract

- Single-layer fringe capacitance using piecewise analysis
- Sidewall capacitance, considering fringing effects

Using Piecewise Analysis

You can use multiple ATTRIBUTE CAP statements to specify piecewise linear coefficients for fringe capacitance based upon the distance between geometries on the same layer. For example

```
PARASITIC CAP[A] METAL3 METAL3 METAL3
ATTRIBUTE CAP[A] 0.8 0
ATTRIBUTE CAP[A] 0.9 0.9
;
; <functions omitted>
;
LPESELECT[C] CAP[A] OUTPUT CDL
```

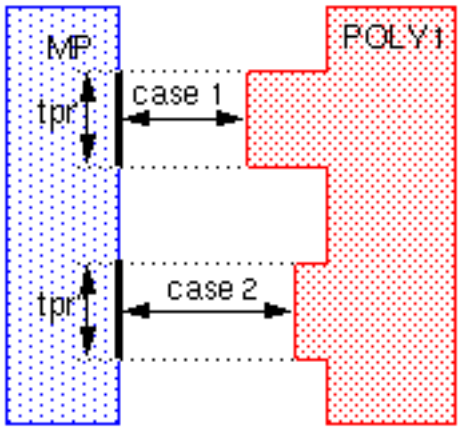
In this example, if the distance between two metal edges is 0 to 0.8, no fringe capacitance is extracted. If the distance is 0.9, the capacitance multiplier is 0.9 per perimeter unit.

Extracting Sidewall Capacitance with Fringing Effects

Piecewise analysis with the ATTRIBUTE CAP statement includes

- Standard overlap and perimeter extraction
- Fringe capacitance based on the separation between the device layer and projecting edges of geometries on the first terminal layer

The following figure shows an example.



You write an `ATTRIBUTE CAP` statement for case 1, where the separation of *POLY1* and *MP* is 0 to 3.0 units. Write another `ATTRIBUTE CAP` statement to cover case 2, where the separation is 3.0 to 5.0 units. The area and perimeter values stay the same for all cases.

```
PARSET = TCAP TPR CLL AREA PERI OVPR C
;
; <functions omitted>
AND POLY1 METAL1 MP
;
; sidewall capacitance with fringing effects
PARASITIC CAP[A] MP POLY1 METAL1
ATTRIBUTE CAP[A] 0.5 1.2  3.0  0.05 ; CLL'=0.05
ATTRIBUTE CAP[A] 0.5 1.2  5.0  0.01 ; CLL"=0.01
;
; <functions omitted>
LEXTRACT TCAP MP POLY1 BY CAP[A] PFILE &
EQUATION C = 0.5 * AREA + 1.2 * (OVPR - TPR) + CLL
;
; <functions omitted>
LPESELECT[C] CAP[A] OUTPUT CDL
```

The `CLL` value is the total fringe capacitance between the two geometries at the given separation, per perimeter unit.

`TPR` is the total length of the perimeter that has fringe effect. For this example, the value of `TPR` is

```
TPR = (tpr' + tpr")
```

and the value of `CLL` is

```
CLL = (0.05 * tpr' + 0.01 * tpr")
```

The capacitance calculated for each capacitor is

```
EQUATION C = 0.5 * AREA + 1.2 * (OVPR - TPR) + CLL
```

This equation subtracts `TPR`, the perimeter used in the fringe effect calculation, from the overlap perimeter, and adds the fringe effect capacitance into the total capacitance value.

Note: This is the default equation that Dracula uses to calculate 2.5D MB parasitic capacitance. You can omit the `PARSET`, `LEXTRACT`, and `EQUATION` functions when they are the same as the default equation.

Using One Function to Extract RC Parasitics

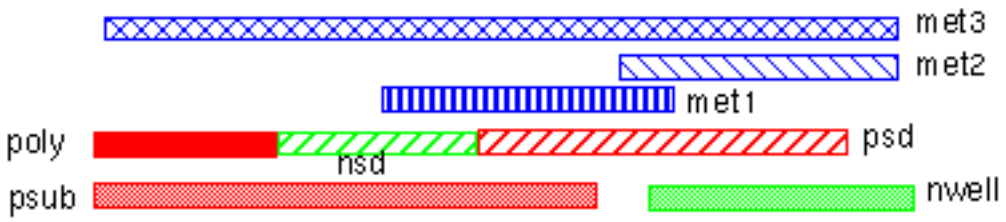
Dracula version 4.5 lets you extract RC parasitics using a single function, `extractParasitic`. Once you specify a list of layers, you can extract overlap, one- and two-layer fringe, piecewise fringe, sidewall, colinear edge, and two-dimensional/three-body (2D3B) capacitance, as well as sheet and contact resistance.

PDRACULA expands the `extractParasitic` function into the necessary capacitance extraction functions and resistance extraction functions and creates a new rules file called *`filename`*`gen.rul`. You can read and edit this file. The operations in this file are executed when you run your extraction job with the `jxrun.com` file.

Defining Parasitics Layers

Before you specify which RC elements to extract with `extractParasitic`, you must include a list of layers to use as capacitor terminals. This example shows the layer list for the following figure.

```
extractParasitic(
     (layers( (nwell psub)(poly psd nsd) met1 met2 met3 ) ...)
```



The *nwell* and *psub* layers are nonoverlapping layers at the lowest level, so they are listed first, grouped in parentheses. The *poly*, *nsd*, and *psd* layers are grouped at the second level. The *met1* layer is at the third level, and *met2* at the fourth level. The *met3* layer is the highest level, and is listed last.

Extracting Overlap and Sidewall Capacitance

After you specify the layer list, you can extract capacitance using the *cap* keyword in the `extractParasitic` function. Use the following syntax to extract simple overlap area and perimeter capacitance.

```
cap( layer1 layer2 area perimeter [lateral(layerN piecewiseList)])
```

The following is an example of extracting simple metal and poly capacitance. This example specifies coefficients for overlap area and perimeter capacitance.

```
;              area  perimeter
cap (BULK POLY   0.35  0.12)  poly to substrate overlap
cap (BULK MET1   0.4   0.15)   metal1 to substrate overlap
```

Extracting Directional Sidewall and Colinear Edge Capacitance

For a more accurate model, you can replace the *perimeter* value in the previous syntax with coefficients for colinear edge, sidewall up, and sidewall down capacitance.

```
cap( layer1 layer2 area colinear sidewallUp sidewallDown
     [lateral(layerN piecewiseList)])
```

The following example, a continuation of the previous example, shows how to extract metal-to-poly directional sidewall capacitance.

```
; directional:   area  colinear sideUp sideDown
     cap (POLY MET2 0.3   0.02     0.1    0.15 )
```

Extracting 2D3B Capacitance

You can use the `lateral` option with the `cap` keyword to extract two-dimensional/three-body capacitance. The following example extracts *POLY*-to-*MET2* capacitance. When *POLY* is separated from the capacitance device layer (the areas where *POLY* and *MET2* overlap) by 1.0 unit or less, the fringe effect of *POLY* on the overlapping layer is represented by a capacitance value of 0.03 per unit of length. If the separation is between 1.0 and 1.5 units, the value is 0.02.

```
cap (POLY MET2 0.2  0.3
     lateral(POLY (1.0  0.03)(1.5  0.02)))
```

Extracting One- and Two-Layer Fringe Capacitance

The `extractParasitic` function has a `fringe` keyword that lets you extract one- and two-layer fringe capacitance using piecewise analysis, as described in "Extracting 2.5D MB Parasitics" on page 117.

```
fringe( layer1 layer2 piecewiseList )
```

The following example shows how to use the `fringe` keyword to extract fringe capacitance.

- For single-layer *MET1* fringe capacitance, when shapes are separated by 1.0 unit or less, the fringe effect is 0.02 per unit of length. If the separation is between 1.0 and 1.5 units, the value is 0.015.
- For *MET1*-to-*MET2* fringe capacitance, when shapes are separated by 1.0 unit or less, the fringe effect is 0.03 per unit of length. If the separation is between 1.0 and 1.5 units, the value is 0.01.

```
fringe(MET1 MET1 (1.0  0.02)(1.5  0.015)) ; one-layer fringe
fringe(MET1 MET2 (1.0  0.03)(1.5  0.01))) ; two-layer fringe
```

Modifying the Coefficient Generator Interface with Dracula to Improve LPE Accuracy

Dracula and Coefficient Generator use the `2D3B` model to calculate the capacitance between two neighboring layers. The current default equation is `C = Ca*AREA + (PERI-TPR)*Cp + CLL`. If one of the layers happens to be a cut-term layer, PERI can include some false edges and overestimate the capacitance. This version of Dracula improves the LPE accuracy by implementing a flexible LPE equation to handle the false edges in the case of neighboring cut-term layers. The enhancement modifies the Coefficient Generator interface with Dracula.

Although the new version of PDRACULA (4.X.0399) still works with the old Coefficient Generator interface, you will get better performance if you use the new version of the Coefficient Generator (1.20-S007). The new version of the Coefficient Generator generates two Dracula rule files instead of one for each Coefficient Generator run. The old interface takes the name that you specify. The new interface will have the extension `.new.` For example, if the Dracula rule file was previously called `dracula.rul,` then two Dracula rule files are generated:

- `dracula.rul,` which contains the Dracula rule in the old format (two capacitances(...) per adjacent layer pair)
- `dracula.rul.new`, which contains the Dracula rules in the new format (one capacitance(...) per adjacent layer pair)

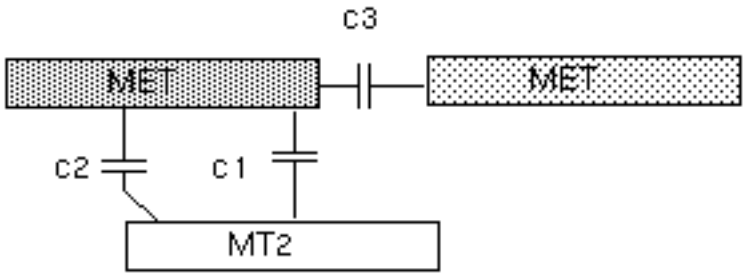
Note: Do not use an old version of PDRACULA with a new version of Coefficient Generator as this will result in inaccurate extractions and must be avoided. To check which version of Coefficient Generator you are using, use the `coeffgen -version` command. For PDRACULA, use the banner during run time for the version information. If you don't use Coefficient Generator to generate the correct `extractParasitic`, the above confusion won't exist as you have provided the coefficients yourself.

New Coefficient Generator and Dracula Interface with the extractParasitic command

The examples below use the following terms:

- *layerList* indicates the layers to be used for capacitor terminal layers.
- *areaCoeff* indicates the coefficient used for overlap area capacitance.
- *periCoeff* indicates the coefficient to be used for perimeter capacitance.
- *fringeCoeff* indicates the coefficient defined in the piece-wise list for capacitance.
- OVPR indicates an overlapped perimeter to the second layer.
- TPR indicates the perimeter of geometries that caused the fringe effect.
- CLL is the sum of the fringe effect related to TPR.

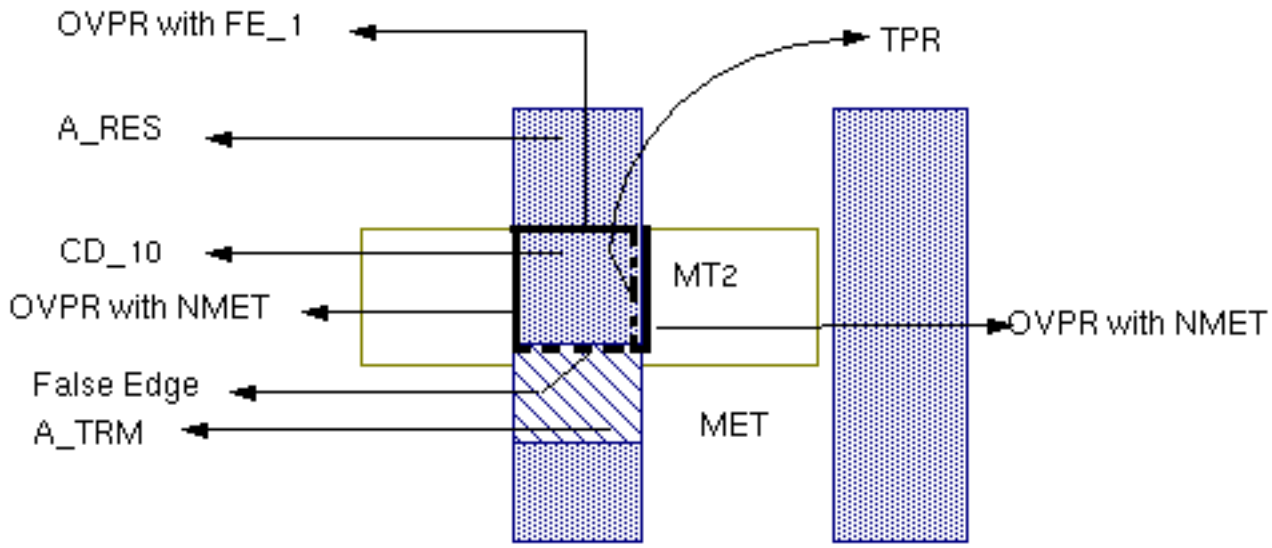
Side View



In the above figure

- c1 stands for the overlap area capacitance
- c2 indicates the perimeter capacitance
- c3 indicates the fringe capacitance

Top View



PDRACULA uses a new format for capacitance commands between adjacent layers. A single capacitance command with two lateral terms is used for adjacent layers to replace two separate capacitance commands, each with a single lateral term. Here is an example.

```
extractParasitic(
    (     layers( layerList )
        cap( MET MT2 areaCoeff periCoeff
            lateral(MET piecewiseList)
            lateral(MT2 piecewiseList) )
        ......................
    )
    (    resistor(MET sheetResValue .......)
        ......................
    )
)
```

The area, perimeter, and fringe coefficients for adjacent layers appear very different from the old Coefficient Generator because in the old Coefficient Generator, the area and perimeter coefficients were divided by two and the fringe coefficients were subtracted by one half perimeter coefficients to avoid double counting. Now, PDRACULA does the necessary adjustments in the expanded rules.

Expanded Rules from the New PDRACULA with the New Coefficient Generator

When at least one of the adjacent layers is a cut-term layer, `C=AREA*areaCoeff+(OVPR-TPR)*periCoeff+CLL` is used to replace the default, `C=AREA*areaCoeff+(PERI-TPR)*periCoeff+CLL`. Note that for each pair of adjacent layers, two capacitance

extractions are needed to take the lateral effect of both layers into consideration. To avoid double counting, the second capacitance has `areaCoeff` set to zero. Here is an example.

```
PARASITIC[R] CAP[A1] A_RES MT2
ATTRIBUTE CAP[A1] areaCoeff 1/2(periCoeff) fringeCoeff ....
PARASITIC[R] CAP[A2] MT2 A_RES
ATTRIBUTE CAP[A2] 0 1/2(periCoeff) fringeCoeff ...
...............
NOT BULK MET NMET
NOT MET MT2 FE_1
LEXTRACT CAPZ  CD_10 NMET BY CAP[A1] pfname &
EQUATION C=AREA*areaCoeff+(OVPR-TPR)*periCoeff+CLL
LEXTRACT CAPZ CD_10 FE_1 BY CAP[A2] pfname &
EQUATION C=(OVPR-TPR)*periCoeff+CLL
...
```

Expanded Rules from the New PDRACULA with the Old Coefficient Generator

For PDRACULA to be compatible with oldere versions of the Coefficient Generator, two LEXTRACTs are used for each capacitance in order to extract OVPR with both upper and lower layers.

```
extractParasitic(
(   layers( layerList )
    cap( MET MT2 1/2(areaCoeff) 1/2(periCoeff)
        lateral(MET piecewiseList) )
    cap( MET MT2 coefficientList
        lateral(MT2 piecewiseList) )
    ........................
    )
    (resistor(MET sheetResValue .......)
    ........................
    PARASITIC[R] CAP[A1] A_RES MT2
    ATTRIBUTE CAP[A1] 1/2(areaCoeff) 1/2(periCoeff)fringeCoeff...
    PARASITIC[R] CAP[A3] MT2 A_RES
    ATTRIBUTE CAP[A3] 1/2(areaCoeff) 1/2(periCoeff) fringeCoeff ....
    NOT BULK MET NMET
    NOT MET MT2 FE_1
    LEXTRACT CAPZ  CD_10 NMET BY CAP[A1] pfname &
    EQUATION CTMP=AREA*1/2(areaCoeff)+(OVPR-TPR)*1/2(periCoeff)
                                                +CLL &
    LEXTRACT CAPZ CD_10 FE_1 BY CAP[A2] &
    EQUATION C=CTMP+(OVPR)*1/2(periCoeff)
```

Older Versions of PDRACULA with Newer Versions of the Coefficient Generator

You must avoid using the older versions of PDRACULA with newer versions of the Coefficient Generator as this combination gives inaccurate results.

Extracting Sheet Resistance

You can extract resistance using the `resistor` keyword in the `extractParasitic` function, as follows:

```
(resistor( resLayer sheetResValue [cont(contactLayerList)]
     [device(deviceLayerList)]
     [maxlength(length)]
     [maxwidth(width)]
     [smash(smashValue maxValue)]
```

The *resLayer*, *contactLayerList*, *deviceLayerList*, *length*, and *width* are the same as defined for CUT-TERM. The *sheetResValue*, *smashValue*, and *maxValue* are the same as defined for ATTRIBUTE RES.

The following is an example of extracting simple poly resistance. In this example, sheet resistance is extracted from the *POLY* conduction layer where it runs between *CONTACT*, *NGATE*, and *PGATE* terminals. The resistance value used is 0.7 per square unit. If a *POLY* resistor is over 30 square units in length, it is cut evenly into multiple resistors.

```
(resistor(POLY 0.7 cont(CONTACT) device(NGATE PGATE) maxlength(30))
```

Because "cont" is a keyword in this syntax, do not use this word as a layer name or PARSET name.

Extracting Multiple Resistance on the Same Metal Layer

In previous releases, you were not able to choose more than one single sheet resistance value for each cut-term layer. The 4.8 version provides a way for Dracula to take multiple sheet resistance values for the same layer based on the layer width. For detailed information about width and sheet resistance criteria, see the "MULTI-SHEETRES" section in the Operation Block Commands Chapter of the *Dracula Reference.*

multiSheetRes in extractParasitic

```
resistor(resLayer sheetResValue [cont(contactLayerList)]
[device(deviceLayerList)]
[multiSheetRes((width1 sheetResValue1)
[(width2 sheetResValue2) [(....)]])
[maxlength(length)]
[maxwidth(width)]
[smash(smashResValue maxResValue)]
[corner45(compenFactor)]
[align]
```

```
)
[resistor....]
```

multiSheetRes in expanded rules

```
PARASITIC RES[A] M1RES M1TRM
ATTRIBUTE RES[A] 0.0008
MULTI-SHEETRES RES[A] width1 sheetResValue1
MULTI-SHEETRES RES[A] width2 sheetResValue2
```

Note: There are certain limitations:

- You use the `MULTI-SHEETRES` command only for parasitic resistance extraction, and this command must follow its corresponding `PARASITIC` and `ATTRIBUTE` commands.
- Flexible LPE is not supported for this version.

Extracting Contact Resistance

The `contact` keyword lets you extract contact resistance. The contact resistance value equals the contact resistance coefficient you supply divided by the contact area (R_c = *coeff / area*).

```
contact( contactLayer layer1 layer2 contactCoefficient)
```

The following example extracts the contact resistance of the *VIA* shapes between the *MET1* and *MET2* layers.

```
contact(VIA MET1 MET2 0.1)
```

Capacitance or Resistor Model in the Extracted Netlist

The usability of the `extractParasitic()` command has been enhanced.

You can can now specify the model type for each capacitance or resistance command in `extractParasitic()`. PDRACULA automatically generates the model commands with the correct subtypes.

The SPICE files will include model types for resistors and capacitors.

Command Syntax

- For `extractParasitic`:

```
extractParasitic(
(   ....................
```

```
        cap( layer1 layer2 coefficientList [model(modelType)]
            [lateral(layer1 piecewiseList)]
        ...................
        fringe(layer1 layer2 piecewiseList [model(modelType)])
            ......................
        )
        ( resistor (layer1 sheetResValue  ... [model(modelType)]
        ( contact (contactLayer layer1 layer2 contactCoefficient
        [model(modelType)])
        )
    )
```

- For the expanded rules of `extractParasitic`:

```
*DESCRIPTION
    MODEL = RES[subType1] modelType1  RES[subType2] modelType2
                                                          ...
    MODEL = CAP[subType3] modelType3  RES[subType4] modelType4
                                                          ...
*END
*OPERATION
.........................
```

Arguments

modelType

The user-specified model type for each capacitor or resistor or contact extraction command. The model option can be placed anywhere after the coefficient list, but before the lateral command for plate capacitors and after the piecewise list or resistance coefficient for resistors or contacts.

Function Syntax

Following is the complete syntax of the `extractParasitic` function.

```
extractParasitic(
     [(extractCapSection)] |
     [(extractResSection)]
     )
```

where

```
extractCapSection ::=
layers( layerList )
cap( layer1 layer2 coefficientList
     [model(modeType)])
     [lateral(layerN piecewiseList
|    fringe( layer1 layer2 piecewiseList)
     [model(modeType)])
```

```
                extractResSection ::=
                 resistor(resLayer sheetResValue
                      [cont(contactLayerList)]
                      [device(deviceLayerList)]
                      [multiSheetRes piecewiseList]
                      [model(modeType)]
                      [maxlength(length)]
                      [maxwidth(width)]
                      [smash(smashResValue maxResValue)]
        |            contact(contactLayer layer1 layer2
                     contactCoefficient [model(modeType)]))
```

Note: So from the above syntax, `extractParasitic` can contain only *extractCapSection* or *extractResSection*. If both exist *extractResSection* must follow *extractCapSection*.

Before You Start

Here are some things to keep in mind when you are coding RC extraction with the `extractParasitic` function:

- You must list the layers used with the `cap`, `lateral`, and `fringe` keywords in the `layers` list. The layers in the `layers` list must be defined in `CONNECT-LAYER` or derived from the layers listed in `CONNECT-LAYER` functions.
- You can use only the conduction layers listed in `CONNECT-LAYER` functions as sheet resistance layers. This means that you cannot use derived layers (layers created by logical or sizing functions) as conduction layers in `extractParasitic` for parasitic resistance.
- You must define contact connections with the `CONNECT` function if you use the `contact` keyword to extract contact resistance.
- You cannot extract sheet resistance alone. You must also extract capacitance. When extracting contact resistance, you must also extract sheet resistance.
- If you are extracting parasitics, put the `extractParasitic` function in the rules file before the `LVSCHK` or `LPECHK` function. If you are doing backannotation, put the `extractParasitic` function in the rules file after the `LVSCHK` or `LPECHK` function.
- You do not need to include `PARSET` definitions for 2D3B capacitance, directional edge capacitance, and contact resistance. The only parasitic extraction functions you need to include are `extractParasitic` and `LPESELECT`.
- The following functions must appear in the Dracula rules file in the order given here. Functions in [ ] are optional.

```
CONNECT-LAYER
[SUBSTRATE]
CONNECT|[SCONNECT]
[STAMP]
[LINK]
[ELEMENT]
extractParasitic()
```

Creating Additional Subtypes

If the `extractParasitic` function does not provide the parasitic capacitance modeling you require, you can define your own subtypes as described in "Using Equations with Flexible LPE" on page 79.

If you define your own subtypes, you must specify extraction functions in your rules file in the following order:

```
PARSET
PARASITIC CAP | RES
extractParasitic()
LEXTRACT
EQUATION
```

The names you give your subtypes must not be the same as the following subtypes that are automatically generated by `extractParasitic`.

- Overlap capacitance

  An uppercase letter and a number: *A0*, *A1*, ... *A9*, *B0*, ... *Z9*.

- Fringe capacitance

  Two uppercase letters: *AA*, *AB*, ... *AZ*, *BA*, ... *ZZ*.

- Resistance

  A single letter followed by an underscore and the layer: *A_TRM* for the terminal layer, *A_RES* for the resistor body layer, *B_TRM*, *B_RES*, ... *Z_TRM*, *Z_RES*.

Examples

There are two CMOS examples in this section:

- Process with two metal layers
- Process with four metal layers

Two Metal Layer Extraction

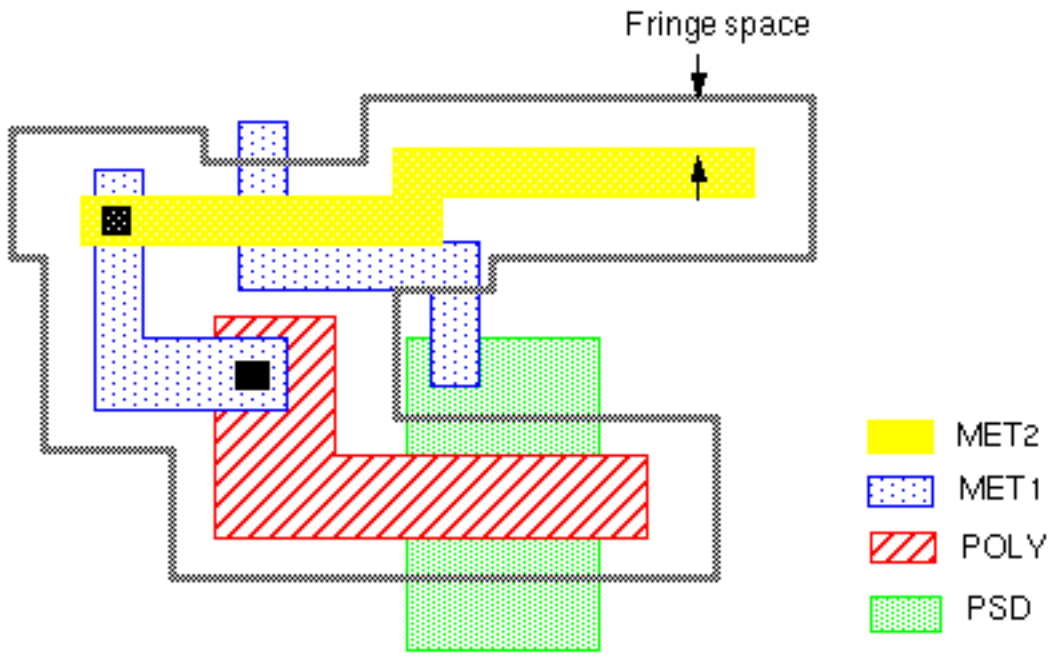
```
; <functions omitted>
;
extractParasitic(
( layers( BULK (NWELL PWELL)(PSD NSD POLY) MET MT2)
fringe(MET1 MET1 (1.0 0.02)(1.5 0.015)) ; 1-layer fringe
fringe(MET1 MET2 (1.0 0.03)(1.5 0.01)) ; 2-layer fringe
cap (BULK POLY 0.35 0.12); poly, substrate overlapped cap
cap (BULK MET1 0.4 0.15) ; met1, substrate overlapped cap
cap (POLY MET1 0.2 0.3 ; 2D3B model for overlapped cap
    lateral(POLY (1.0 0.03)(1.5 0.02)))
cap (POLY MET2 0.3 0.02 0.1 0.15))     ; sidewall cap
( resistor(MET1 0.1 cont(CONTACT VIA))
resistor(MET2  0.3 cont(VIA) maxlength(30))
resistor(POLY 0.7 cont(CONTACT) device(NGATE PGATE))
contact(VIA MET1 MET2 0.1))
)
; <functions omitted>
LPESELECT[S] CAP &
LPESELECT[S] RES OUTPUT SPICE
```

Four Metal Layer Extraction

```
; <functions omitted>
;
extractParasitic(
(layers(SUB MET1 MET2 MET3 MET4)
cap(MET1 SUB 25.4E-6 42.0E-6
    lateral(MET1 (0.79 0.0)(1.2 8.9E-6)(1.6 5.91E-6)(2.0 4.41E-6)))
cap(MET2 SUB 13.8E-6 37.7E-6
    lateral(ME2 (0.89 0.0)(1.3 5.1E-6)(1.7 3.54E-6)(2.25 2.71E-6)))
cap(MET3 SUB 9.3E-6 34.4E-6
    lateral(MET3 (0.89 0.0)(1.3 3.1E-6)(1.7 2.15E-6)(2.25 1.65E-6)))
cap(MET4 SUB 6.9E-6 35.6E-6
    lateral(MET4 (2.49    0.0)(3.0 12.6E-6)(3.5 10.4E-6)(4.0  9.14E-6)(4.5  7.88E-
6)
    (5.0 6.93E-6)(5.5 6.30E-6)(6.25 5.67E-6)))
cap(MET2 MET1 48.0E-6 51.9E-6
    lateral(MET2 (0.5 33.7E-6)(0.9  24.8E-6)(1.3 19.6E-6)(1.7 15.9E-6)
    (2.25 13.7E-6)))
cap(MET3 MET1 18.1E-6 40.6E-6
    lateral(MET3 (0.5 12E-6)(0.9  8.85E-6)(1.3 7.0E-6)(1.7 5.68E-6)(2.25 4.89E-
6)))
cap(MET4 MET1 11.0E-6 40.1E-6
    lateral(MET4 (0.5 59.3E-6)(0.9 43.7E-6)(1.7 34.5E-6)(2.5 24.1E-6)(4.1 18.9E-6)
    (5.3 13.0E-6)(6.5 10.4E-6)))
cap(MET3 MET2 49.7E-6 53.2E-6
    lateral(MET3 (0.5 35.9E-6)(0.9  26.4E-6)(1.3 20.9E-6)(1.7 17.0E-6)
    (2.25 14.6E-6)))
cap(MET4 MET2 18.2E-6 46.0E-6
    lateral(MET4 (0.5 87.2E-6)(0.9 64.2E-6)(1.7 50.8E-6)(2.5 35.5E-6)
    (4.1 27.8E-6)(5.3 19.2E-6)(6.5 15.3E-6)))
cap(MET4 MET3 49.7E-6 63.1E-6
    lateral(MET4 (0.5 160.4E-6)(0.9 118.6E-6)(1.7 93.4E-6)(2.5 65.2E-6)(4.1 51.1E-
6)
    (5.3  35.2E-6)(6.5 28.2E-6)))
fringe(MET1 MET1 (0.79 61.1E-6)(2.00 0.00))
fringe(MET2 MET2 (0.89 64.8E-6)(2.25 0.00))
fringe(MET3 MET3 (0.89 63.9E-6)(2.25 0.00))
fringe(MET4 MET4 (2.49 27.9E-6)(6.25 0.00))
)

(resistor(MET1 87.0E-6 cont(VI1) maxlength(2500))
resistor(MET2 66.0E-6 cont(VI1 VI2) maxlength(2500))
resistor(MET3 66.0E-6 cont(VIA2 VIA3) maxlength(2000))
resistor(MET4 39.0E-6 cont(VIA3) maxlength(1000))
contact(VIA1 MET1 MET2 2.5E-3)
contact(VIA2 MET2 MET3 2.5E-3)
contact(VIA3 MET3 MET4 2.5E-3)
)
)
;
LPESELECT[S]  CAP GT 0.0 &
LPESELECT[S]  RES GT 0.0 OUTPUT SPICE
```

Extracting a Single Net

If you need to extract RC elements for a single net only, you can use the `SMART-LPE` function in Dracula version 4.5. This function selects a net for which Dracula extracts RC parasitics, including the fringe capacitance within a space around the net. PDRACULA looks through all the fringe-related definitions in your rules file and calculates a fringe space for each connection layer accordingly.



Note: If you are using single-net extraction because the volume of RC data for a whole circuit is too large, you might consider using the Cadence® RC network reducer option (formerly called ConcICe), the interconnect reducer, to reduce RC output. For more information about this product, see *ConcICe Help*.

Before You Start

Here are some things to keep in mind when you are writing a rules file to extract a single net with the `SMART-LPE` function:

- You must place all parasitic extraction functions (layer operations) in the rules file after the `ELEMENT` definitions.

- You must place all operations that generate layers used as device layers before the `CUT-TERM` or `extractParasitic` function for a PRE layout-only job.

Coding Single-Net Extraction

To extract a single net, you must include the following functions in your rules file in addition to connection and extraction functions:

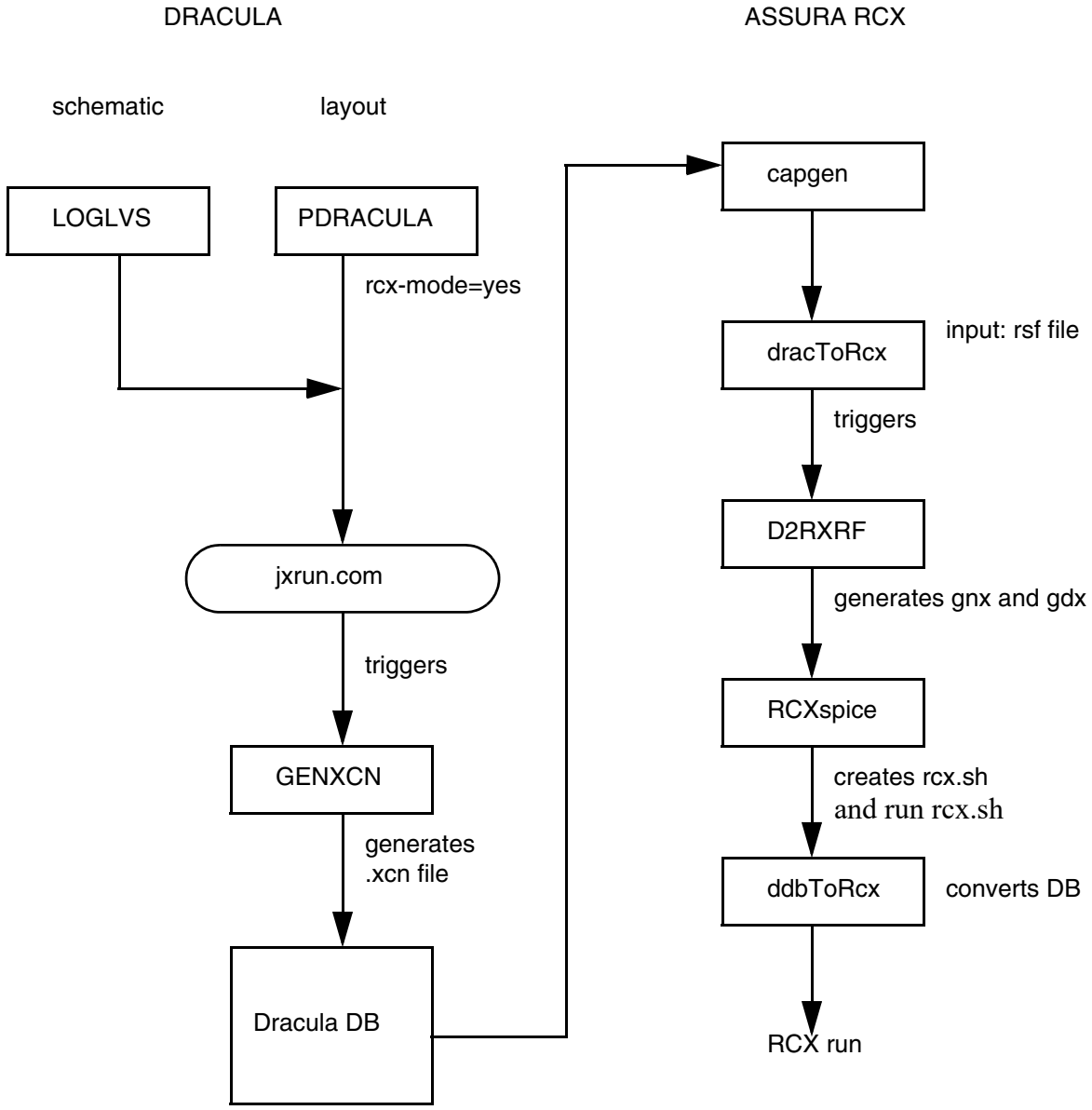
```
*DESCRIPTION
     LPE-QUERY = CORE | EXPAND
     SMART-LPE = SINGLENET
     SELECT-MODE = SCH | LAYOUT filename
     ;
     ; <functions omitted>
     ;
     *END


*OPERATION
     NODE-SELECT = nodename | NODE-FILE = filename
     ;
     ; <functions omitted>
     ;
     *END
```

The Dracula To RCX (dracToRcx) Interface

The Dracula to Assura RCX flow (dracToRcx) provides an alternate solution to the original Dracula PRE flow. dracToRcx allows you to use Dracula LVS as your verification tool for LVS and to then use Assura RCX as your extraction tool, well known for its accurate and sophisticated extraction of interconnect parasitic devices.

The following diagram illustrates the dracToRcx flow.

dracToRcx Flow



Important

The `PRECISION` command is not supported in the dracToRcx flow, and so the environment variable `LVS_PRECISION_3` should be used in place of the `PRECISION` command. Please refer to the *Dracula Reference* manual for more information.

Module Descriptions

Five new modules have been developed for the dracToRcx flow. The new modules are GENXCN, capgen, D2RXRF, ddbToRcx, and dracToRcx, and each is described below.

GENXCN

The GENXCN module prepares an `.xcn` file for capgen. The `.xcn` file contains information such as text, layers, connectivity element names, and model names and is derived from the rule file. Below is an example of a typical `.xcn` file:

```
...
connect tpdiff psub by 0pplug
connect tndiff nxwell by 0nplug
model=mos[p],p
model=mos[n],n
model=mos[pd],pd
model=mos[nd],nd
model=mos[y],y
stamp coll by tpdiff
stamp emit by tpdiff
stamp pdio by pdiff
stamp ndio by ndiff
element mos[n] mosxn8g cpoly ndiff psub
element mos[p] mosxp8g cpoly pdiff nxwell
element mos[y] mosxy8g cpoly yndiff psub
element mos[nd] mosxndg cpoly bndiff psub
element mos[pd] mosxpdg cpoly bpdiff nxwell
...
```

Note: Use the SUBTYPE-CSENS = YES command to maintain the case sensitivity of the device model names.

capgen

capgen is used to perform capacitance pre-characterization from a process file and also to generate capacitance models for RCX. The contact layer names in the `xcn` file are slightly different from those in the rule file. Depending on the situation, a number might appear in the prefix of a contact layer. This difference, illustrated in the following example, must be taken into consideration while preparing a `p2lvsfile` for capgen.

rule file	xcn file
connect mt2 mt1 by via	connect mt2 mt1 by 0via

NOTE: For extraction of capacitors, if scientific notation is used to specify capacitance in the Dracula rule file, the "-cap_unit 1"option should be used while running capgen to prevent RCX from rescaling. For example, in the following rule file

```
ELEMENT        CAP[C2]    DPCAP    FGATE    POLY
PARAMETER      CAP[C2]    0.71E-15
```

capacitance per unit area is specified as 0.71E-15, and "-cap_unit 1"option should be turned on for capgen. For more information, please refer to the RCX chapters of the *Assura Command Reference.*

D2RXRF

This module generates `gnx` and `gdx` files. Both files are in plain text format; `gnx` contains the mapping of net names between layout and schematic while `gdx` contains the mapping of device names.

After a Dracula LVS run is completed, the files `6NXRF.DAT` and `6EXRF.DAT` are created. Each file contains the cross-reference information between layout and schematic, `6NXRF.DAT` for nets and `6EXRF.DAT` for devices. Both files are in Dracula DB format.

Using D2RXRF, the files are translated into plain text files. The net/device name mapping can then be used in the output spice file. If the net names from the layout side need to be kept, you must use the command `SCH-NETNAME=NO` in the rule file to keep schematic net names from being used in the output spice file.

In the `gnx` and `gdx` files, the first line is a count of the items mapped. The second line defines the primary cell name for both schematic and layout; schematic name is on the left and layout name is on the right. The names are separated by a colon, and each record in the `gnx` file

begins with an equal sign (=) while each record in a `gdx` file begins with a ‘”D“. The following are typical `gnx` and `gdx` files:

```
gnx
# 5
topcell: TOPCELL
= VSS VSS
= VDD_0_ VDD_0_
= O_9_ O_9_
= O_8_ O_8_
= O_7_ O_7_
gdx
# 3
topcell: TOPCELL
D M1 drDEV_1
D M2 drDEV_2
D M3 drDEV_3
```

ddbToRcx

The main function of ddbToRcx is to transform the Dracula DB to Assura RCX.

dracToRcx

This utility controls the Dracula to AssuraRCX flow, creates the script, and invokes the necessary processes for a dracToRcx run. The input for the dracToRcx utility is an RCX command file, illustrated as follows:

```
*RCX_DESCRIPTION

       working-directory            = ./

       technology                   = 24-xxx25lo_122502_3D_av2

       tech-lib                     = ./tech.lib

       lvs-source                   = dracula

       dracula-run-name             = A_lvs

       dracula-run-dir              = ./

       cell-name                    = A_lvs
```

```
        run-name                        = runrcx
        type                            = full
        extract                         = cap
        cap-extract-mode                = coupled
        cap-ground                      = GND
        cap-coupling-factor             = 1.0
        net-name-space                  = schematic
        output-format                   = spice
        output                          = runrcx.sp
        max-fracture-length             = infinite
        fracture-length-units           = microns
        tempdir                         = ./tmp
        cap-models                      = comment
        parasitic-cap-models            = comment
        res-models                      = comment
        parasitic-res-models            = comment
        output-net-name-space           = schematic
        sub-node-char                   = #
        hierarchy-delimiter             = /
        toggle-xgl                      = t
        parasitic-res-width             = nil
        extract-mos-diffusion-res       = t
        xy-coordinates                  = C,c
*END
```

RCX commands must be in a block beginning with "*RCX_DESCRIPTION" and ending with "*END". run-name is used to define the name of the subdirectory under the working directory for a particular run.

In the example shown above, the subdirectory runrcx is created under the working directory., after which dracToRcx will generate the script "rcx.sh" in that directory.

dracToRcx will then create processes to run D2RXRF and RCXspice to create `gnx` and `gdx` files, and to prepare other RCX-related files. Then, `rcx.sh` will be executed to create an output spice file specified as the `output`. In the above case, the output spice filename is `runrcx.sp`.

RCX Commands

As mentioned above, commands in RCX command files should be put in the block embraced by *RCX_DESCRIPTION and *END. Also, the commands are case sensitive. The following table gives the description and arguments of each command:
:

command	description
black-box-cells (file_name)	Specifies a set of cells which will be treated as black boxes.
cap-coupling-factor (factor)	Decoupled capacitors will be multiplied by the specified factor.
cap-extract-mode (coupled | decoupled | decoupled_to_substrate)	Parasitic capacitors will either be coupled, decoupled to nearest substrate node, or decoupled to the -cap_ground net (default) when the output netlist is produced.
cap-ground (net)	Decoupled parasitic capacitors will be lumped to the specified net.
cap-models (yes | no | comment)	Models for canonical capacitors will either be written to the output file (default), not written, or written as comments.
cell-name (cell_name)	Top cell name for the LVS run
dracula-run-dir (directory)	Specifies the directory where Dracula LVS is run.
dracula-run-name (run_name)	Specifies the run name of Dracula LVS.
exclude-floating-nets (t | nil)	This option will perform floating net removal via the -rmfloat option to the xreduce module.
extract (cap | res | both | RLC)	Parasitic capacitors (default), resistors, both capacitors and resistors, or RLC’s will be extracted.
extract-mos-diffusion-ap (t | nil)	Causes diffusion area and perimeter parameters AS, AD, PS, PD to be extracted for MOSFETs and LDDs.

import-globals (t | nil)	Specifies that global power and ground nets declared in the LVS input data (such as "`power:P`" and "`gnd:G`") are imported into Assura RCX

To enable a command of Boolean type, set the command to `t` in the command file. To disable the command, set it to `nil`. The command type can be found by using the command "RCXspice -args".

For more information, please refer to the RCX chapters of the *Assura Physical Verification Command Reference*.

Running the DracToRCX Interface

Prerequisites

The following are required to run DracToRCX.

- Dracula version 4.8.1.0903 or above
- Assura
  - AV3.0 (AV3.0.6 or above)
  - AV3.1 (AV3.1.1 or above)
- The following programs in the Dracula/Assura hierarchy (placed automatically):
  - D2RXRF
  - GENXCN
  - ddbToRcx
  - dracToRcx

Running DracToRCX

Following are the steps required to run DracToRCX.

1. Enter `RCX-MODE=YES/LAYOUTONLY` in the description block of the Dracula rule deck to enable DracToRCX flow.
2. Enter `KEEP-SHORT`*`XXX`*`=YES` commands in the description block to keep shorted devices, where *`XXX`* represents MOS, RES, CAP, BJT, or DIO.

3. Use M-Factor commands in the rules file to process M factor correctly. The following are Dracula M-Factor commands:

```
MOS-M-FACTOR = WIDTH-LENGTH/NO
BJT-M-FACTOR = WIDTH-LENGTH/EMITTER-AREA/NO
RES-M-FACTOR = WIDTH-LENGTH/VALUE/NO
DIO-M-FACTOR = AREA/PERI/NO
CAP-M-FACTOR = AREA/VALUE/NO
```

   For more information on M-Factor commands, see "M-Factor Commands" in the "Description Block Commands" chapter of the *Dracula Reference Manual.*

4. Use PDRACULA to compile the Dracula rule deck and to create the run script, `jxrun.com`

5. Run the script `jxrun.com`. This will create an `.xcn` file.

6. Prepare an RCX command file (for example, "`rcx.rsf`") with the switch `lvs-source` set to "dracula" and `output-format` set to "spice", and define the `dracula-run-dir` and `dracula-run-name`. You must also define the same name for both the `cell-name` in the RCX command file and for the `PRIMARY` command in the Dracula rules file. For example:

```
cell-name = VB_1703B_SCH (RCX command file)
PRIMARY = VB_1703B_SCH (Dracula rules file)
```

   Note: The cell name is case sensitive, so the names must match exactly. See the *Dracula Reference* manual for more information on the `PRIMARY` command.

7. Run capgen by using the `.xcn` file created in step 5 as the lvs file. This will convert the lvs extraction rules to rcx extraction rules. For a given process, capgen only needs to be run once. One doesn't have to rerun capgen for each dracToRcx run. For more information about running capgen, see "Capgen Overview" in the "CAPGEN and Assura RCX" chapter of the *Assura Physical Verification Developer Guide*, version 3.0.

8. Run dracToRcx as follows:
   % dracToRcx rcx.rsf

You are now ready to use the data generated by Dracula LVS to run extraction using Assura RCX. For more information on running Assura RCX, see the "RCX Run Guide" chapter of the *Assura Physical Verification User Guide.*

Note: To perform RC extraction more accurately, it is strongly suggested that the diffusion layer be separately defined between the source/drain area and the substrate in the rules file. Also, the contact among metal1-poly, metal1-source/drain and metal1-substrate should be separately defined.

Tip

To allow DracToRCX to support layout-only jobs, specify `RCX-MODE=LAYOUTONLY` instead of `RCX-MODE=YES` in the rules file. Also, use the `LPESELECT` command in the rule file to invoke LVSNET stage in jxrun.com so that 6NETLRCX.DAT will be created.

Tip

To prevent schematic net names from being used in the output SPICE file, use `SCH-NETNAME=NO` in the DESCRIPTION BLOCK in the rule file. The default value of `SCH-NETNAME` is set to YES.

6

Setting up Hierarchical Dracula

In this chapter, you'll learn about the following:

- About Hierarchical Dracula
- Running a Hierarchical Design Rule Check
- Selecting Hcells
- Running a Multilevel Design Rule Check
- Running a Hierarchical Electrical Rules Check
- Running a Hierarchical Network Comparison
- Running Hierarchical Parameter Extractions
- Making Hierarchical Dracula Run Faster

About Hierarchical Dracula

You can use the Dracula hierarchical tools (HDRC, HERC, HLVS, HLPE, and HPRE) on hierarchical designs in the same way you use the corresponding tools on flat designs. The basic differences are as follows:

- You must include hierarchical functions in your rules file.
- Dracula organizes your output differently, depending on which check mode you select.

In the sections that follow, you will learn

- Which Dracula functions you need in your rules file to run a hierarchical job
- How you can interpret the output from a hierarchical job

For more information about when to use hierarchical Dracula, see Chapter 4, "Selecting a Run Mode."

Prerequisites

To run hierarchical Dracula, you must know how to run the Dracula verification tools in flat mode.

Also, you need to be familiar with how Dracula forms a two-level hierarchy consisting of the composite and Hcell planes. For a complete description of how Dracula constructs its hierarchy, see "Hierarchical Structure" in the "Overview" chapter of the *Dracula Reference.*

Hierarchical Products and Modes

This table shows the products you can run in hierarchical Dracula and the modes you can use to run them. For more information about these modes, see Chapter 4, "Selecting a Run Mode."

CHECK-MODE=	HDRC	HERC	HLVS	HLPE	HPRE
HIER	**x**				
CELL	**x***	**x**	**x**	**x**	
COMP	**x***	**x**	**x**	**x**	**x**
MULTI	**x**				

Running only cell and composite mode HDRC is not recommended because these modes do not provide Hcell-to-Hcell or Hcell-to-composite checks. Use hierarchical mode instead.

Running a Hierarchical Design Rule Check

You use hierarchical Dracula in the same way you use flat Dracula to check your design rules. When you run a hierarchical design rule check (HDRC) job, Dracula verifies your database completely. Hierarchical Dracula checks cells with highly repetitive placements only once, and then checks them against the surrounding geometries. The hierarchical output is easier to debug than flat output because errors are reported only once for all placements of a given cell.

You might need to use some or all of the hierarchical functions listed in the following table. To find more information about using the function, look in the Description column of the table.

Writing Hierarchical Design Rules

The rules file you use for a hierarchical DRC run is essentially the same as a rules file for a flat run, with the addition of several hierarchy-related functions. The following table lists the hierarchical functions that you might want to include. For more information about these functions, see the *Dracula Reference* manual.

You cannot use the LENGTH, EXT[G], and RELOCATE functions in hierarchical rules files. Also, you must use flat layers for nodal functions, as described in the "Using Flat Layers for Nodal Information" section

.

Hierarchical Functions for HDRC

Function	Description	Rules file block
CELL-ERROR-REP	Specifies the format of cell-based error segments generated by HDRC; ONCE, HIER, ALL, ORIG-ALL, ORIG-DRC, or cellname. See Using HDRC Output.	*DESCRIPTION
CELLBOX-LAYER	Creates cell boundaries for each Hcell on the layer you specify. See Using Cell Boundaries.	
CHECK-MODE	Set to HIER for HDRC.	*DESCRIPTION
ENVIRONMENT-MAX	Specifies the upper limit for the width of the Hcell environment. See Specifying a Maximum Environment Value.	
EXCEPTION-ON	Set to [HSP-OUTPUT] and use with SIZE to output SIZE results hierarchically. This option also disallows overlapping Hcells during automatic Hcell selection. See Preventing Hcell Overlap.	
HCELL	Specifies which cells are Hcells. See Altering the Hcell Selection Criteria.	
HCELL-FILE	Specifies a file containing a set of HCELL statements. See Using a Cross-Reference File for Hcell Names.	
HCELL-HEIGHT-LIM	YES sets a limit of 1000 microns for the maximum Hcell height. NO specifies that there is no height limit.	

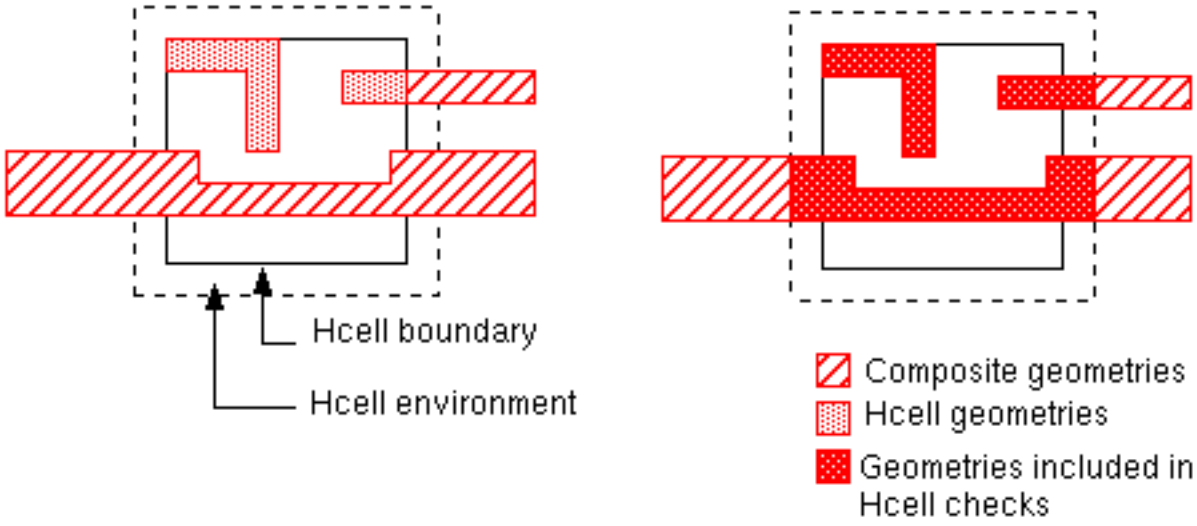
Hierarchical Functions for HDRC, *continued*

Function	Description	Rules file block
HCELL-MAX-SEGMENTS	Specifies the maximum number of line segments for a cell to be an Hcell candidate, to avoid selecting large cells.	*DESCRIPTION
HCELL-MAX-PLACEMENTS	Specifies the maximum number of placements for a cell to be an Hcell candidate, to avoid selecting small cells with a large number of placements.	
HCELL-RULE	Specifies the minimum number of placements and segments for a cell to be an Hcell candidate.	
NOT-HCELL	Specifies cells that cannot be Hcell candidates.	
CELLBNDY	Specifies the digitized layer that contains rectilinear cell boundaries. See Using Cell Boundaries.	*INPUT-LAYER
FLATTEN	Creates a flat layer from a hierarchical layer. See Flattening Layers.	*OPERATION
HIERARCHEN	Creates a hierarchical layer from a flat layer.	

Specifying a Maximum Environment Value

When you run HDRC in hierarchical mode, Dracula surrounds every Hcell with a box that extends beyond the Hcell for a given distance. This distance is called the “environment value.” Dracula automatically sets the environment value to be equal to the largest value used in any spacing check in the Operation block on data that has not been flattened.

Composite geometries that fall within the environment area are checked as part of the Hcell, as shown:



The larger the environment value, the larger the amount of composite data that is checked for each Hcell instance.

➤ To have PDRACULA warn you if the environment value exceeds the value you specify, include the `ENVIRONMENT-MAX` function in the Description block.

For example:

```
ENVIRONMENT-MAX = 10 MIC
;
; <functions omitted>
;
EXT      METAL           LT  5.0      OUT      METERR55
ENC      CONT    METAL   LT  1.5      OUT      CONERR56
EXT      POLY    DIFF    LT  3.0      OUT      PLYERR57
EXT      PAD     METAL   LT  30.0     OUT      PADERR58
```

The default environment value for this job is 30 microns. In this case, Pdracula warns you that the 30 micron spacing check exceeds the maximum environment value of 10 microns.

In response to the warning, you can do one of the following:

- Leave the environment set to the default largest size. Dracula will take extra time mapping additional data to the Hcells.
- Flatten the layers used in the 30 micron check. For more information, see the "Flattening Layers" section.

If the ENVIRONMENT-MAX value you set is larger than the largest spacing check value, Dracula reduces the environment value to the lower spacing check value.

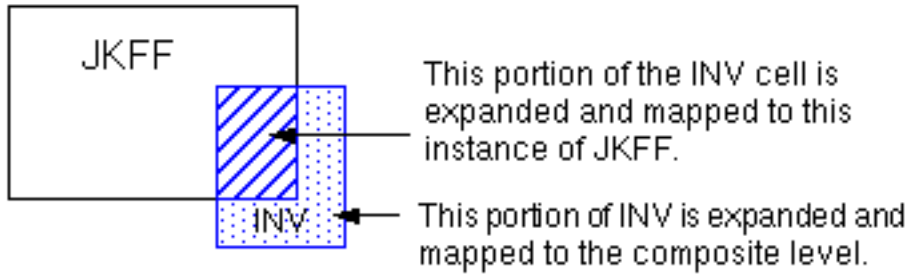
Note: The ENVIRONMENT-MAX function works with HDRC hierarchical mode only.

Caution

If you set ENVIRONMENT-MAX to 1 micron and you do a 3-micron spacing check, Dracula can miss errors. An error that is 2 microns out is not included in the environment.

Preventing Hcell Overlap

While selecting Hcells automatically, Dracula might choose overlapping cells. If the amount that two Hcells overlap is greater than half the area of either of the Hcells, Dracula chooses only one of the cells as an Hcell. Generally, Dracula chooses the cell with the least amount of overlapped data.



Dracula expands cells that do not qualify as Hcells and brings them up to the composite level. If the cells are nested in an Hcell, Dracula brings them up to the Hcell level and maps them to that instance of the Hcell.

Some designs run more efficiently in hierarchical mode if the Hcells Dracula uses do not overlap.

➤ To prevent Dracula from selecting a large number of overlapping Hcells in your design using the default criteria, put the following function in the Description block of your rules file:

```
EXCEPTION-ON = [HSP-OUTPUT]
```

This function prevents Dracula from choosing as an Hcell any cell that overlaps another cell.

You can also include `EXCEPTION-ON = [HSP-OUTPUT]` when you use the hierarchical `SIZE` function, so the results will be clean and correct.

Flattening Layers

The `FLATTEN` function creates a new flat layer at the top level from a hierarchical layer. It is useful only in hierarchical mode. In flat mode, the `FLATTEN` function uses the UNIX `cp` command to copy data from one file to another file with a different name.

You might need to use `FLATTEN` in your HDRC rules to do

- Large spacing checks
- Checks that require nodal information

You can also use `FLATTEN` in HLPE to extract capacitors between an internal cell node and a parent cell. See the "Flattening Layers To Extract Capacitance" section.

Note: You can use both hierarchical and flat layers in the Operation block of an HDRC rules file, but you cannot combine hierarchical and flat layers in a single check.

Avoiding Flat Processing

If you want to take full advantage of the hierarchy of your circuit when running HDRC, you can merge any operations that require flat layers into an ERC job rather than including them in an HDRC job.

Large Spacing Checks

Dracula uses the spacing value for hierarchical checks to determine the environment value, as described in the section about Specifying a Maximum Environment Value. However, Dracula does not include checks on flat layers when determining this value.

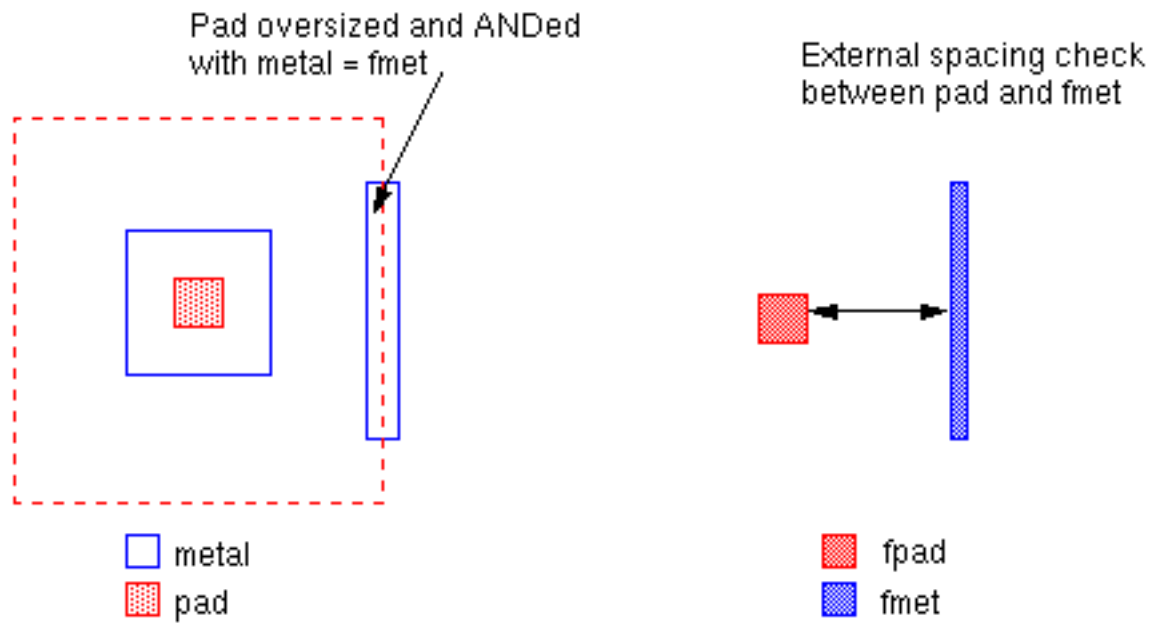
➤ To do checks with large spacing values without affecting the environment value, use the `FLATTEN` function.

Pad-to-metal checks normally use large spacing values and involve many geometries. To avoid affecting the environment value, use the `FLATTEN` function on the pad and metal layers. To reduce the amount of data that you must process, which in turn reduces the processing time, use logical operations.

The following example creates a layer, *PMET*, consisting of metal geometries no farther than 30 microns from the pad geometries. Then *PAD* and *PMET* are flattened and used in the 30 micron spacing check.

```
; The NEIGHBOR function does these two operations:
; SIZE PAD BY 30 SPAD
; AND SPAD METAL PMET
;
NEIGHBOR METAL PAD 30 PMET
;
; <functions omitted>
;

FLATTEN PAD FPAD
FLATTEN PMET FMET
EXT[V] FPAD FMET LT 30 OUT PADERR 58
```

With these functions, you can perform relatively few large dimensional checks without affecting the performance of the entire HDRC run.



Using Flat Layers for Nodal Information

You must use flat layers for the following HDRC checks and operations that require nodal information:

- `EXT[N]`
- `ENC[N]`

- `INT[N]`
- `CONNECT`
- `SELECT` *`layer`* `LABEL`
- `STAMP`
- `LINK`

For information on different flattening requirements for HDRC and HERC, see "Connecting the Network" section.

Checking Memories and Arrays

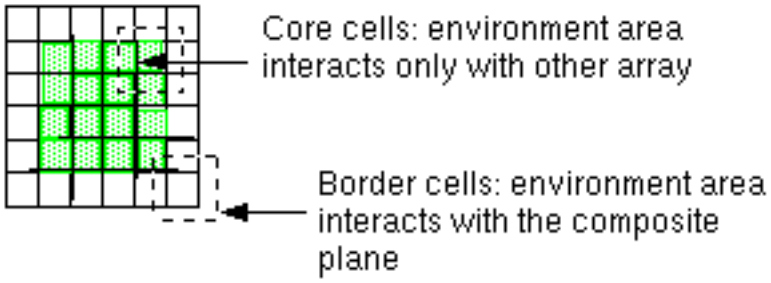
To process large arrays and memories, you can use the following HDRC features:

- Multilevel mode

  If your designs have repetitive structures with many levels of hierarchy, such as large DRAMs, you can use multilevel mode for HDRC runs. For more information about using this mode, see Using Multilevel Mode in Chapter 4 of this manual.

- `RAM-CELL` function

  To reduce redundant checking, you can use the `RAM-CELL` function in the Description block of your rules file. When Dracula sets up a database for HDRC, it creates an environment area for each cell, as described in Specifying a Maximum Environment Value. In an array of cells, the environment areas of the cells around the border of an array are different because they interact with the composite plane. The environment areas of the core cells, however, are all the same. The following figure illustrates an array of cells.



The Hcell you specify for the `RAM-CELL` function must be a repetitive cell with simple, nonrotated placements. Composite geometries cannot overlap the core Hcells in the array.

Using HDRC Output

You view and correct errors from an HDRC run the same way you do for a flat run. The error cells can contain Hcell and composite geometries. Dracula organizes the content of the cells according to the option you specify with the `CELL-ERROR-REP` function. These options are described later in this section.

Viewing Hcell Error Cells

Hcell error cells contain errors associated with Hcells and any pseudolayers created by logical, `SIZE`, or `SELECT` operations that occur for all placements of the Hcells.

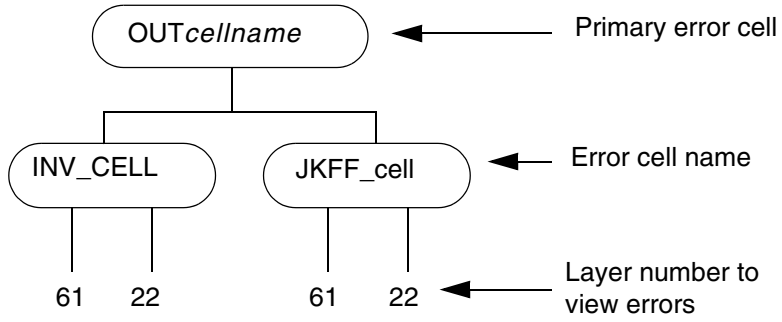
Dracula generates Hcell error cells when you run an HDRC in either of the following modes:

- Cell mode (`CHECK-MODE = CELL`)
- Hierarchical mode (`CHECK-MODE = HIER`) with `CELL-ERROR-REP = HIER`

The following sample circuit contains two Hcells, INV and JKFF. There are two HDRC checks in the rules file:

```
EXT  METAL LT  5  OUT METERR 61
SIZE POLY  BY .25 OUT SPOLY  22
```

Hcell error cells from these checks appear as shown:



Dracula uses the layer numbers you give for output error layers to isolate the error within the Hcell output. Each output cell contains error flags on layer *61* and sized data on layer *22*, if applicable. The origin of these error cells is in relation to the origin of the corresponding layout cell.

Using Error Reporting Options

You can use the `CELL-ERROR-REP` options to report errors using two different hierarchy structures:

- Original design hierarchy

  The `ORIG-ALL`, `ORIG-DRC`, and *`cellname`* options report errors using the original hierarchy of your design.

- Two-level hierarchy

  The `ONCE`, `HIER`, and `ALL` options report errors using the two-level hierarchy (composite plane, Hcell plane) that Dracula constructs. These options are described in the sections that follow.

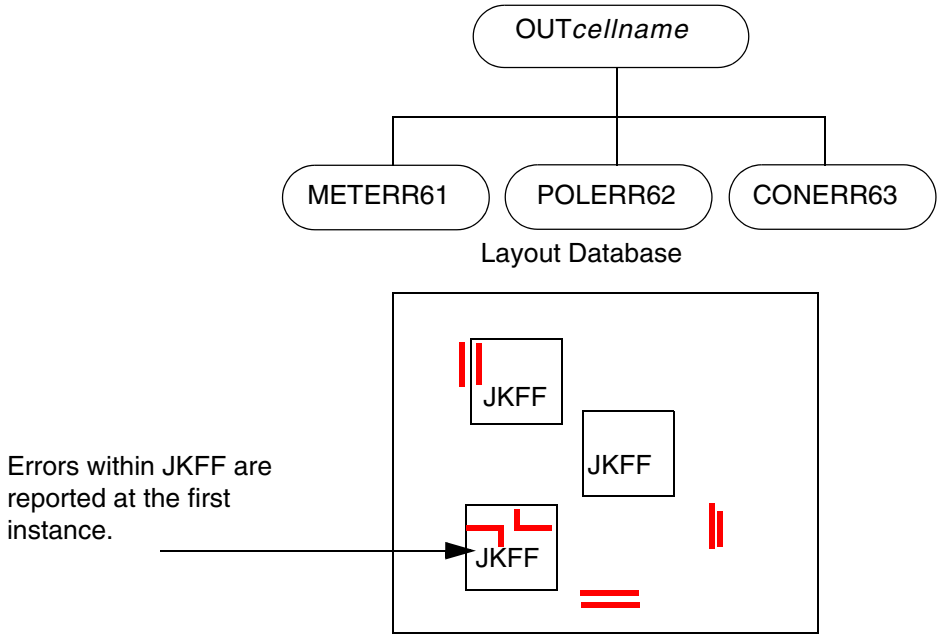
For more information about these options, see the section about "CELL-ERROR-REP" in the "Description Block Commands" chapter of the *Dracula Reference.*

Viewing a CELL-ERROR-REP = ONCE Error Cell

When you use `CHECK-MODE=HIER` with `CELL-ERROR-REP = ONCE` in your rules file, you get an error cell for each check. These cells contain errors that occur

- Between two Hcells
- Between an Hcell and its environment
- Between composite-level geometries that are not Hcell-related
- At every placement of an Hcell

These errors are shown in one Hcell. This Hcell is the one closest to the minimum x/y coordinate of the primary cell.



Viewing a CELL-ERROR-REP = ALL Error Cell

The error cells produced when you set `CELL-ERROR-REP=ALL` in your rules file are the same as the cells produced by a flat Dracula run.

Viewing a CELL-ERROR-REP = HIER Error Cell

When you use `CHECK-MODE=HIER` with `CELL-ERROR-REP = HIER` in your rules file, you get

- Check error cells

  Dracula produces an error cell for each check. Each check error cell contains errors that occur between two Hcells or between an Hcell and its environment, but do not occur at each placement of the Hcell. Each cell also contains errors that occur between composite-level geometries that are not Hcell-related. These errors are reported flat and relative to the primary cell.

- Hcell error cells

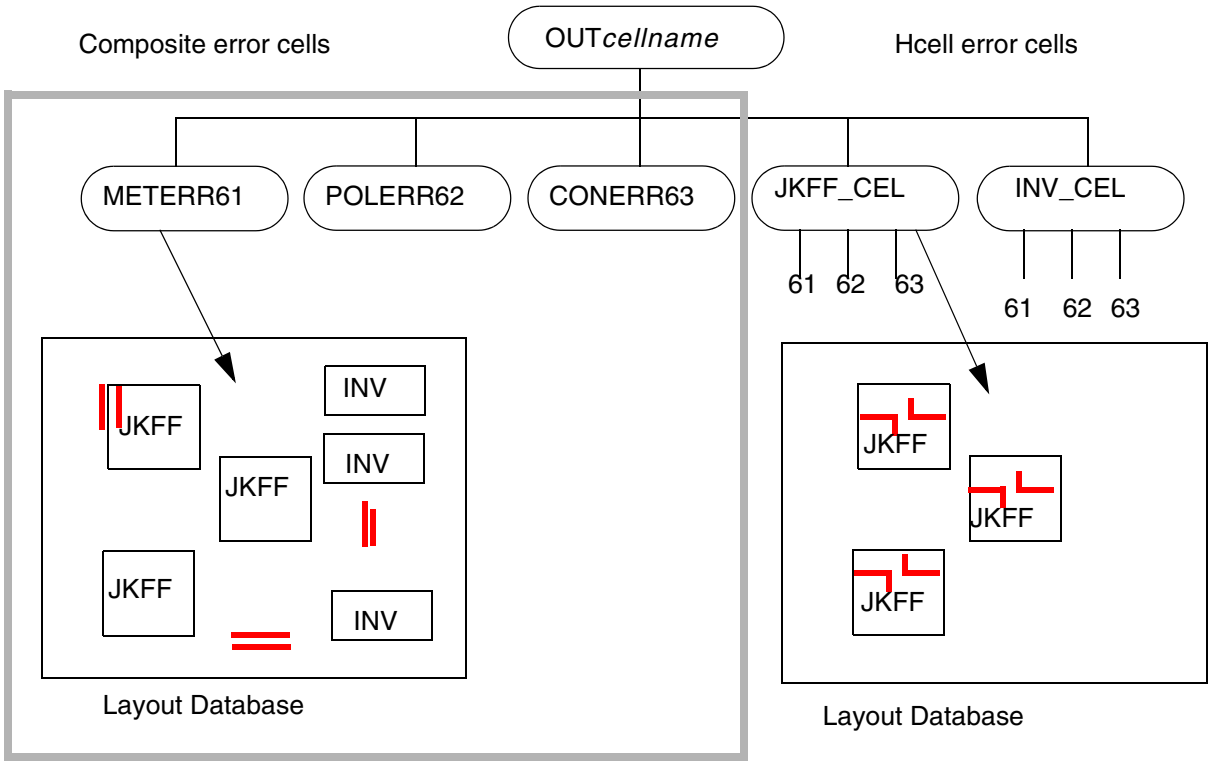
  Dracula produces an Hcell error cell for each Hcell. Each Hcell error cell contains errors that occur at every single placement of the Hcell. To see these errors at each placement

of the Hcell, overlay the primary error cell, OUT*primarycellname*, on the original database. To see a single instance, overlay the Hcell error cell on the original Hcell database cell.

Here is an example of checks in a rules file:

```
EXT METAL      LT 5 OUT METERR 61
EXT POLY       LT 3 OUT POLERR 62
ENC CONT METAL LT 2 OUT CONERR 63
```

These rules produce the following HDRC error cells, which you can overlay on both the Hcell and the top-level design:



Listing HDRC Text Output Files

HDRC text output includes the following files:

- *printfile*.sum, the error cell summary
- *printfile*.log, the job tracking log
- *printfile*.mlg, the mini-log file with CPU statistics and fatal errors

- *printfile*.msm, the mini-summary file with nonfatal errors and warnings
- *printfile*.inp, a listing of the input rules file

where *printfile* is the name you specify with the PRINTFILE function in the Description block of your rules file.

For a complete description and examples of HDRC output, see "HDRC Error Reports" in the "Checking Design Rules (DRC)" chapter of the *Dracula Reference.*

Selecting Hcells

You usually let Dracula select Hcells automatically by running HDRC in hierarchical mode first. If the results are not satisfactory, you might want to choose your own Hcells.

To select the Hcells to use in hierarchical Dracula jobs, you need to be familiar with how Dracula forms its two-level hierarchy by creating the composite and Hcell planes. For a complete description of how Dracula chooses Hcells, see "Hierarchical Structure" in the "Overview" chapter of the *Dracula Reference.*

Selecting Hcells Automatically

For hierarchical and multilevel HDRC, Dracula uses default selection criteria to select Hcells automatically. There is one set of criteria for hierarchical HDRC and a different set for multilevel HDRC. You can find a description of these criteria, along with instructions for changing or overriding them, in the "Selecting HDRC Hcells" in the "Checking Design Rules (DRC)" chapter of the *Dracula Reference.*

Note: If you are running HDRC, but you did not include an HCELL function in your rules file and Dracula cannot identify any Hcells, Dracula automatically switches to flat mode.

Run HDRC with the default Hcell selection first, particularly if you are not familiar with the layout hierarchy. You can look at the following files to determine which cells Dracula chose to be Hcells:

- The CANDIDAT.DAT file lists the Hcells that Dracula chose:

```
1  INV       2          1          5 0          20
2  JKFF1     8          6          9 0
```

- The EXPAND stage of the .log file lists Hcell selection statistics:

```
*****************************************************
*/N*    EXPAND   ( REV. 4.3         /      SUN-4
GENDATE: 12-APR-95/13 )
*** ( Copyright 1995, Cadence ) ***
```

```
*/N*     EXEC TIME =15:37:49       DATE = 2-MAY-95
HOSTNAME = xxxxxx
*****************************************************


RUN-MODE   =     3
TREE-FILE  = TREEGRAY

5.2500    0    0    0    1    0    0    0    0

****    ENVIRONMENT =     5.2500   *****



*****  HCELL-SELECT-CONDITION  *****

# OF PLMTS         # OF LINE-SEGMENTS
       2             44

# OF CELLS        :         8
# OF PLACEMENTS   :        29

PRIMARY CELL    : GRAYCODE5

NUMBER OF H-CELLS =          2
:
UPDATED NUMBER OF HCELLS =          2

NUMBER OF BOXES       =         13
NUMBER OF CELL PLMTS =         13

NO CELL PLACEMENT WITH MAGNITUDE < > 1
:
*******   CHECK BOX OVERLAP   *******

INDIVIDUAL CELL STATISTICS         8         8

             # OF       # OF ORIGINAL
            PLACEMENTS PLACEMENTS   LEVEL KIDS HCELL
1 GRAYCODE5 0          1            2     28   0
2 INV       5          9            1     0    1
3 NO2       0          1            1     0    0
4 PAD       0          11           1     0    0
5 ND4       0          1            1     0    0
6 ND3       0          1            1     0    0
7 ND2       0          1            1     0    0
8 JKFF1     4           4            1     0    2



        # OF     # OF
         LINE-SEGSTEXTS ******** CELL BOUNDARY ********
```

```
1 GRAYCODE57765 0         0     -1890     20060     12235
2 INV       142 0         0         0      2200       840
3 NO2       214 0         0         0      2200      1120
4 PAD        12 0         0         0      1200      1540
5 ND4       274 0         0         0      2200      1680
6 ND3       214 0         0         0      2200      1120
7 ND2       175 0         0         0      2200       840
8 JKFF1    1180 0         0         0      2940      7260

*/M** TOTAL NUMBER OF EDGES IF FLATTENED =      7765
      NUMBER OF EDGES IN H-CELLS         =      5430
      H-CELL COVERAGE RATE               =     69.93%
      FULL HIERARCHICAL EDGES            =      2969
      2-LEVEL HIERARCHICAL EDGES         =      3657

# OF CELLS        :         8
# OF PLACEMENTS   :        29
ORIGINAL # OF PLMTS BRANCHES =         29
NEW      # OF PLMTS BRANCHES =         33
:
---------------------- HCELL CANDIDATES ---------------

1  INV                        2            1           5
2  JKFF1                      8            6           9

-----------------------------------------------------------
```

Selecting Hcells Manually To Improve Performance

If your circuit performs poorly in hierarchical mode, you might be able to improve performance by choosing different Hcells.

General Guidelines

Look for Hcells that have all of these qualities:

- Repetitive

  The Hcell should be placed at least four times (more is better).

- Complex

  The best candidate for an Hcell contains about 5% of the total design.

- Nonoverlapping

  The Hcell must not overlap with other cells or noncell geometries.

The challenge is to find Hcells that are both complex and repetitive. If the Hcell is too simple, most of the processing time is spent on bookkeeping. A contact cell is an example of a poor choice for an Hcell. The cell is repeated many times, but is too simple to improve the processing time for a hierarchical check.

Try to pick Hcells that are near the middle of the hierarchy tree. The cells at the bottom of the hierarchy might have many instances, but they are very simple. Cells near the top of the hierarchy are complex, but may only be placed a few times.

Choose few Hcells rather than many. For memory-based designs, for example, a single Hcell is the best choice.

Getting Started

1. Start with the basic `HCELL-RULE` function.

   If you have never selected your own Hcells, the following is a good rule to start with. The rule specifies that, to qualify as an Hcell, a cell must be placed more than 5 times if it contains more than 1000 segments, or more than 10 times if it contains more than 500 segments.

   ```
   HCELL-RULE 5,1000 10,500
   ```

2. Check the log file for the Hcell area coverage rate.

   After you run Dracula using these criteria, look at the `EXPAND` stage of the `.log` file. This part of the file tells you how much of the total area of your design is covered by Hcells that were selected using your criteria. In the following example, the coverage rate is 86.08%.

   ```
   */M** TOTAL NUMBER OF EDGES IF FLATTENED =      7673
         NUMBER OF EDGES IN H-CELLS         =      6605
         H-CELL COVERAGE RATE               =      86.08%
         FULL HIERARCHICAL EDGES            =      2959
         2-LEVEL HIERARCHICAL EDGES          =      3039
   ```

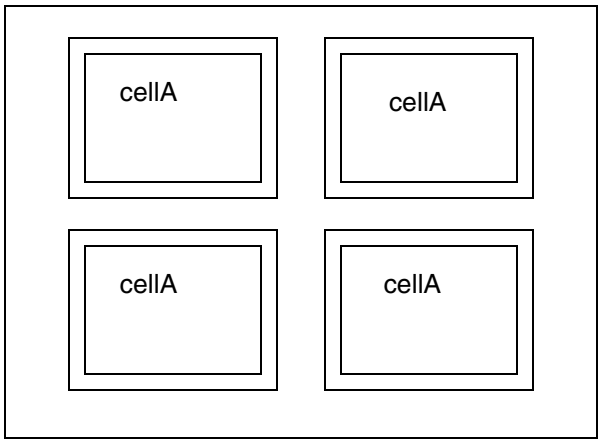
   Try to pick Hcells to increase the coverage rate and decrease the number of Hcells listed in the `CANDIDAT.DAT` file. You want the number of Hcells listed to be less than 200.

Running a Multilevel Design Rule Check

For designs on which the standard two-level HDRC does not run as efficiently as you want, you can use multilevel HDRC. Memory chips where overlap is minimal are the best candidates for multilevel HDRC, as described in “Using Multilevel Mode” on page 62.

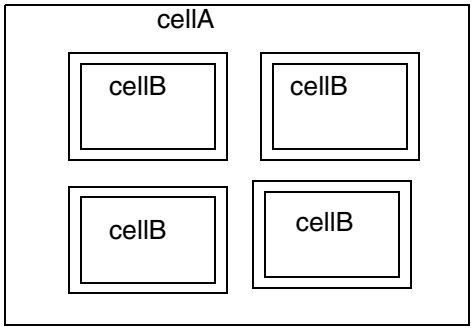
Understanding Multilevel Processing

To process a multilevel design, Dracula creates multiple two-level hierarchies, starting at the top level. The following figure shows the first level of a multilevel design. Dracula creates the composite layer on the top level. It creates frames for the Hcells and checks the composite layer and its connections to the Hcell frames.



The size of the Hcell environment area is the largest spacing value in a hierarchical DRC check (see Specifying a Maximum Environment Value). However, if there are intrusions into the Hcell, the spacing value is increased.

When Dracula has checked the top level, it goes to the next level, which is the *cellA* Hcell in the previous example. Treating *cellA* as the top level, Dracula again creates a composite plane with frames around the Hcells that *cellA* contains. Dracula continues down the hierarchy in this fashion.



Running a Multilevel Hierarchical DRC

You can use an HDRC rules file for multilevel HDRC. See Running a Hierarchical Design Rule Check.

To run a multilevel HDRC, see "Multilevel Mode" in the "Checking Design Rules (DRC)" chapter of the *Dracula Reference.*

Running a Hierarchical Electrical Rules Check

To check the electrical parameters of a circuit, you use hierarchical Dracula in the same way you use flat Dracula. You verify your cells prior to verifying the top-level design. Then, when you verify the top-level design, Dracula treats cells as blocks with pin names only, and checks only the external connections to other blocks.

Writing Hierarchical Electrical Rules

The rules file you use for an HERC run is the same as the rules file for a flat run, with the addition of several hierarchy-related functions. The following table lists the Dracula hierarchical functions that you might want to include. For more information about these functions, see the *Dracula Reference Manual*.

In cell or composite modes, HERC functions can be run with HLVS and HLPE functions in the same rules file, because they use the same mechanism for defining Hcell and cell text

.

Hierarchical Functions for HERC

Function	Description	Rules file block
`CELL-CHILD-TEXT`	Cell mode only. For a description of this function, see Bringing Text Up to the Hcell Level. (Not recommended — causes HERC errors.)	`*DESCRIPTION`
`CELLBOX-LAYER`	Creates cell boundaries for each Hcell in the `OUTDISK` file. See the section about Using Cell Boundaries.	
`CHECK-MODE`	Set to `CELL` or `COMPOSITE` for HERC.	`*DESCRIPTION`

Hierarchical Functions for HERC, *continued*

Function	Description	Rules file block
FLATTEN-PWRGND	Composite mode only. For a description of this function, see Flattening Power and Ground Text.	
HCELL	Specifies which cells are Hcells. Mandatory for HERC. (FRAME BY option not recommended — causes HERC errors.) See Altering the Hcell Selection Criteria.	
HCELL-FILE	Specifies a file containing a set of HCELL statements. See Using a Cross-Reference File for Hcell Names.	
BASE-LAYER	Specifies layers containing composite-level geometries that must be mapped into Hcells. See Using Composite Plane Geometries in Hcells.	*INPUT-LAYER
CELLBNDY	Specifies the digitized layer that contains rectilinear cell boundaries. See Using Cell Boundaries.	*INPUT-LAYER
CTEXT	Specifies the input layer number of Hcell text.	
HEDTEXT	Specifies the file containing Hcell text.	*OPERATION
FLATTEN	Creates a flat layer from a hierarchical layer.	
XBOX	Checks for composite geometries on the specified layer that touch or overlap Hcell bounding boxes.	
XCELL	Checks for geometries from the layer1 composite plane that overlap Hcell layer2 geometries. Checks for geometries from the layer2 composite plane that overlap Hcell layer1 geometries.	
XDEVICE	Outputs composite-to-cell and cell-to-cell devices that are not present in the schematics. Composite mode only.	

Hierarchical Functions for HERC, *continued*

Function	Description	Rules file block
XVIA	Checks for contacts that are not covered by a related conductor layer in either the composite plane or Hcells.	*OPERATION

Deciding What To Check

When you run HERC in cell mode, Dracula verifies only the Hcells in your database.

When you run HERC in composite mode, Dracula does the following:

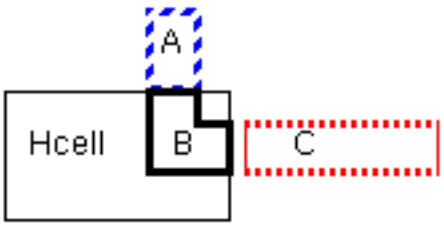
- Checks the interconnect between Hcells
- Checks cells that are not defined as Hcells
- Runs SAMELAB, MULTILAB, SOFTCHK, ELCOUNT, and PATHCHK checks on the Hcell contents

For a complete verification, you must run both cell mode and composite mode HERC. All checks that you run in composite mode must be the same checks you ran in cell mode.

Connecting the Network

Although you need to flatten layers to do nodal checks in HDRC, you do not need to flatten the layers you use in CONNECT functions for HERC and HLVS.

HDRC doesn't use Hcell text labels. In the following figure, *A*, *B*, and *C* are all on the same node. The B geometry in the Hcell has text labels on both sides of the Hcell boundary. In a composite mode HDRC, however, Dracula ignores the boundary text and treats *A* and *C* as two different nodes, unless you flatten the layers.



HERC and HLVS require and work with labelled Hcell boundaries. For HERC and HLVS, Dracula makes the connection from the text on the Hcell boundary. HERC/HLVS treat *A*, *B*, and *C* as one node, without requiring flat layers.

Flattening Power and Ground Text

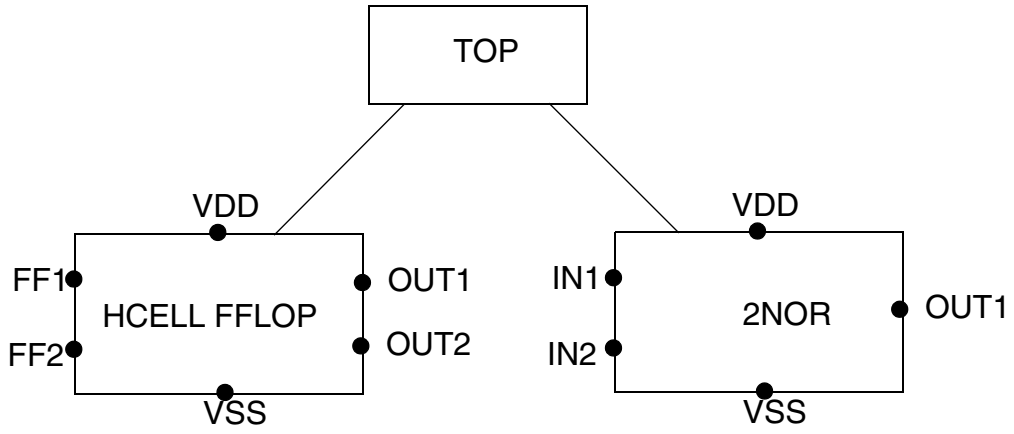
You can flatten the power and ground text from the Hcell level to the composite level. This feature is useful in the following cases:

- You are running a composite mode HERC and you have not texted the top-level structure
- You want additional text on the composite plane

➤ To flatten power and ground text to the composite level, include this function in the Description block of your Dracula rules file:

```
FLATTEN-PWRGND = YES
```

In the following example, *VDD* and *VSS* from both Hcells are expanded up to the composite level for use with composite mode HERC. If there is a power or ground short from the `MULTILAB` function, the output error cell, which lists the shortest distance between two texted points, might be shorter because you have more text. See Viewing the SHORTBOX Cell.



Excluding Hcell Geometries in HERC

Although you can use the `FRAME BY` option with the `HCELL` function to remove Hcell geometries from processing, there are reasons to avoid this option in HERC:

- If cell text is placed within the area of the Hcell that is excluded from processing, the text is also excluded, so nodal information is not available.

- If you have routing over your Hcells, shorts between composite nodes and nodes inside the core of the Hcell are not flagged because the core is excluded from processing.

The `FRAME BY` option causes false HERC errors and is not recommended for an HERC run.

Using Composite Plane Geometries in Hcells

In hierarchical Dracula, you can form devices solely within Hcells or solely on the composite plane, but not between the composite plane and Hcells. However, you can use the `BASE-LAYER` function to include portions of composite geometries in your Hcells to form devices.

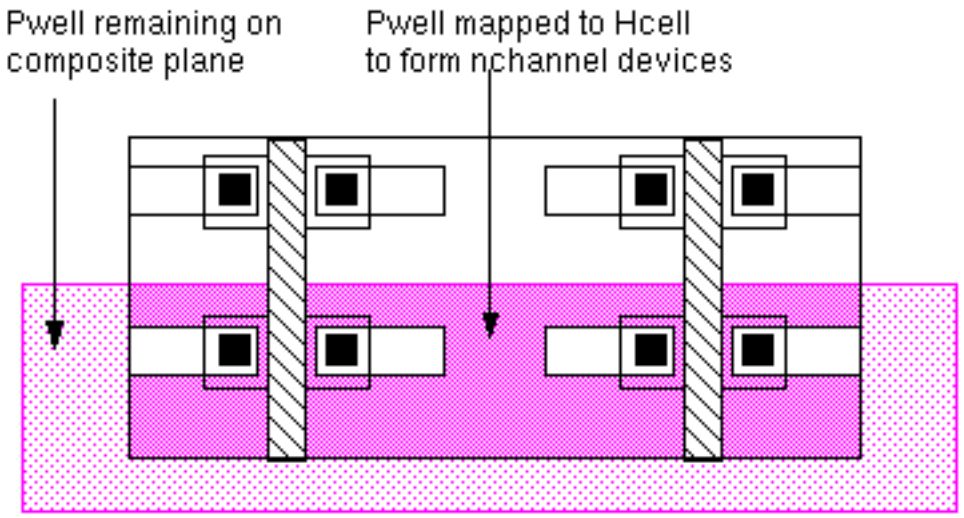
You might want to create a flat layer on an otherwise hierarchical circuit. For example, you can create cells in a pwell CMOS process that have no pwell layer. Then, after you create the entire circuit, you can create the pwell layer as a flat layer where needed. You need to map the flat or composite pwell geometries to the Hcell so that the Hcell is complete and n-channel devices can be formed in the Hcells.

➤ To correctly map composite-plane geometries on a layer to Hcells, use the `BASE-LAYER` function.

For example:

```
*INPUT-LAYER
ACTIVE = 1
POLY   = 2 TEXT = 61 ATTACH = METAL
METAL  = 3
PWELL  = 6
BASE-LAYER = PWELL
CONNECT-LAYERS = NSUB PWELL PSD NSD POLY METAL
* END
*OPERATION
;
; <functions omitted>
;
ELEMENT MOS[N] NGATE POLY NSD PWELL
*END
```

You need the pwell layer to form the body of n-channel devices in the Hcells. Dracula includes overlapping portions of the pwell layer from the composite plane in the Hcells, as shown in the following figure.



If you use the `BASE-LAYER` function in cell mode, you must also use it in the subsequent composite mode run to ensure proper verification.

For a complete description of this function, see the section about the "BASE-LAYER" function in the "Input-Layer Block Commands" chapter of the *Dracula Reference.*

Using HERC Output

You view and correct errors from an HERC run the same way you do for a flat run. The error cells can contain Hcell and composite geometries.

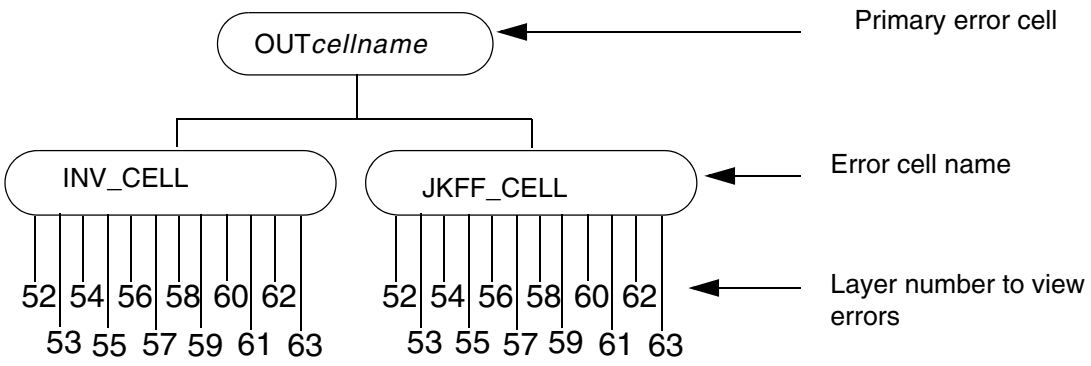
Viewing Cell Mode Error Cells

A sample circuit contains two Hcells, *INV* and *JKFF*. The rules file contains this HERC check:

```
MULTILAB OUTPUT MULLAB 52
```

When you include checks that have nodal-type output, such as `MULTILAB`, Dracula produces multiple layers for that check. For example, if your design has 7 connect layers and 4 contact layers, Dracula outputs 11 different layers to carry the nodal information. The *INV_CEL* and

JKFF_CEL output cells in the following figure contain layers *52* to *62*, which all have geometries from the *MULTILAB* check.



To find out what the output layers are, look in your *printfile*.erc file MULTILAB summary. The layer numbers start at the layer number you specify for the MULTILAB output cell, and increment by one. In the following example, the output layer number the user specified was 40. There were 3 contact layers specified in CONNECT functions and 6 connection layers specified in the CONNECT-LAYER function. The last layer is the SHORTBOX cell, described in Viewing the SHORTBOX Cell.

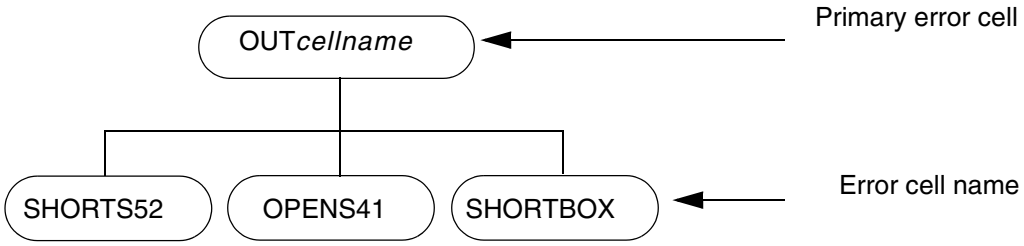
```
***** MULTILAB SUMMARY *****

0 TRAPEZOIDS ARE SELECTED FROM 3METAL   TO SHORTS40 AT LAYER   40
0 TRAPEZOIDS ARE SELECTED FROM 3POLY    TO SHORTS40 AT LAYER   41
0 TRAPEZOIDS ARE SELECTED FROM 3PSD     TO SHORTS40 AT LAYER   42
0 TRAPEZOIDS ARE SELECTED FROM 3NSD     TO SHORTS40 AT LAYER   43
0 TRAPEZOIDS ARE SELECTED FROM 3PWELL   TO SHORTS40 AT LAYER   44
0 TRAPEZOIDS ARE SELECTED FROM 3NSUB    TO SHORTS40 AT LAYER   45
0 TRAPEZOIDS ARE SELECTED FROM 030CONT  TO SHORTS40 AT LAYER   46
0 TRAPEZOIDS ARE SELECTED FROM 030NTAP  TO SHORTS40 AT LAYER   47
0 TRAPEZOIDS ARE SELECTED FROM 030PTAP  TO SHORTS40 AT LAYER   48
0 TRAPEZOIDS ARE SELECTED FROM SHORTBOX TO SHORTS40 AT LAYER   49
```

Viewing Composite Mode Error Cells

For the following HERC rules

```
MULTILAB OUTPUT SHORTS 52 1
SAMELAB  OUTPUT OPENS  41 2
```

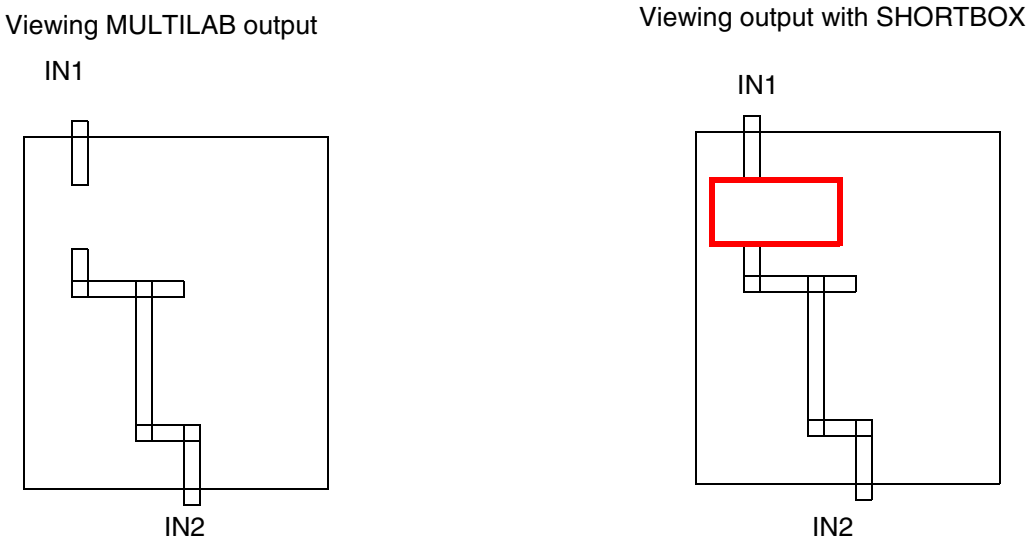
the composite error cells for HERC are organized as follows.



The results of `ELCOUNT` and `PATHCHK` on the Hcell plane are passed to the pins of the Hcells. Any composite node connecting to those Hcell pins combines the pin information with the composite node.

Viewing the SHORTBOX Cell

The `MULTILAB` error output shows the composite geometries that represent the shortest path between two texted points where a short occurred. When you include a `MULTILAB` check in your HERC rules file, Dracula creates a cell called `SHORTBOX`. The `SHORTBOX` cell contains the cell bounding boxes through which the composite node passed. Overlay this cell on with your `MULTILAB` error cell to locate errors, as shown in the right side of the following figure.



Listing HERC Text Output Files

HERC text output includes the following files:

- `printfile.sum`, the error cell summary
- `printfile.erc`, the HERC summary report containing `SOFTCHK` results, text information, shorts, opens, and error cell summaries
- `printfile.log`, the job tracking log
- `printfile.mlg`, the mini-log file with CPU statistics and fatal errors
- `printfile.msm`, the mini-summary file with nonfatal errors and warnings
- *`printfile`*`.inp`, a listing of the input rules file

where *`printfile`* is the name you specify with the `PRINTFILE` function in the Description block of your rules file.

For information about HERC output, see "HERC Commands" in the "Checking Electrical Rules (ERC)" chapter of the *Dracula Reference.*

Running a Hierarchical Network Comparison

To completely verify the connectivity of a circuit, you use hierarchical Dracula in the same way you use flat Dracula. You verify your cells prior to verifying the top-level design.

For more information about HLVS, see "Hierarchical LVS" in the "Comparing Layouts and Schematics (LVS)" chapter of the *Dracula Reference.*

Writing Hierarchical Network Comparison Rules

The rules file you use for an HLVS run is the same as a rules file for a flat run, with the addition of several hierarchy-related functions. The following table lists the Dracula hierarchical functions that you might want to include in your rules file for an HLVS run. For complete documentation about these functions, see the Dracula Reference.

Note: You cannot form devices by overlapping two Hcells. You can form elements in the Hcells or on the composite plane, but not between a combination of the two. The BASE-LAYER function is the only exception to this rule

.

Hierarchical Functions for HLVS

Function	Description	Rules file block
CELL-CHILD-TEXT	Cell mode only. For a description of this function, see Using Cell Mode in Chapter 4.	*DESCRIPTION
CELLBOX-LAYER	Creates cell boundaries for each Hcell. See the section about Using Cell Boundaries.	
CHECK-MODE	Set to CELL or COMPOSITE.	
GEN-TEXT-FILE	Instructs Dracula to automatically text Hcells and place the text in a HEDTEXT file. See *"Using GEN-TEXT To Hcell Text"*.	
GEN-TEXT-FLTNODE	YES assigns text to feedthroughs and floating nodes. NO disables this option.	
GEN-TEXT-FRAME	Generates a frame inside Hcell boundaries for automatic texting.	
GEN-TEXT-WIRE	YES automatically generates internal-wire and must-join wire types. NO disables this option.	*DESCRIPTION
HCELL	Specifies which cells are Hcells. See "*Altering the Hcell Selection Criteria*".	
HCELL-COLUMN-1	SVS only. Specifies whether the first column in the HCELL-FILE lists layout or schematic cells.	
HCELL-FILE	Specifies a file containing a set of HCELL statements. See Using a Cross-Reference File for Hcell Names.	
HCELL-HEIGHT-LIM	YES sets a limit of 1000 microns for the maximum Hcell height. NO specifies that there is no height limit.	

Hierarchical Functions for HLVS, *continued*

Function	Description	Rules file block
HCELL-MAX-SEGMENTS	Specifies the maximum number of line segments for a cell to be an Hcell candidate, to avoid selecting large cells.	
HCELL-MAX-PLACEMENTS	Specifies the maximum number of placements for a cell to be an Hcell candidate, to avoid selecting small cells with a large number of placements.	*DESCRIPTION
BASE-LAYER	Specifies layers containing composite-level geometries that must be mapped into Hcells. See *"Using Composite Plane Geometries in Hcells"*.	*INPUT-LAYER
CELLBNDY	Specifies the digitized layer that contains rectilinear cell boundaries. See *"Using Cell Boundaries"*.	
CTEXT	Specifies the input layer number of the Hcell text.	
GEN-TEXT-LAYER	Cell mode only. See Specifying Texting Layers in Chapter 4 of this manual.	
HEDTEXT	Specifies the name of the Hcell text file.	*OPERATION

For complete verification, you must run both cell mode and composite mode HLVS.

Note: Each Hcell must have a corresponding .SUBCKT definition in a SPICE/CDL netlist. In composite mode, only the I/Os are required, not the transistors. All I/Os defined in the netlist must be texted on the Hcell with the same name. For more information about texting Hcells, see Texting for Cell Mode in Chapter 4 of this manual.

HLVS Prerequisites

See "Hierarchical LVS" in the "Comparing Layouts and Schematics (LVS)" chapter of the *Dracula Reference* for the following information:

- A list of the files you need to run HLVS
- A complete listing of an HLVS rules file

You might want to create separate composite- and cell-mode rules files, because some functions might be different between the two files.

Note: GEN-TEXT-FILE, HEDTEXT, and CTEXT/ATTACH can be used in both files, depending upon your application.

Functions for Cell Mode HLVS

Type	Function	Reference
mode	CHECK-MODE = CELL	None
Hcells	HCELL *layout netlist*	See Using a Cross-Reference File for Hcell Names.
texting	GEN-TEXT-FILE = *filename*	See *"Using GEN-TEXT To Generate Hcell Text"* in Chapter 4 of this manual.
	layername = # CTEXT = # ATTACH = *layer*	

Functions for Composite Mode HLVS

Type	Function	Reference
mode	CHECK-MODE = COMP	None
Hcells	HCELL *layout netlist* FRAME BY #	See *"Excluding Hcell Geometries in HLVS"*.
texting	HEDTEXT = *filename*	See *"Using GEN-TEXT To Generate Hcell Text"* in Chapter 4 of this manual.

Note: GEN-TEXT-FILE, HEDTEXT, and CTEXT/ATTACH can be used in both files, depending upon your application.

Excluding Hcell Geometries in HLVS

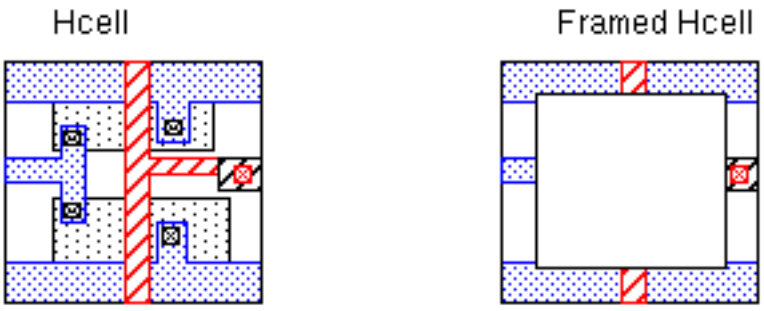
When you use the FRAME BY option with the HCELL function for composite-mode HLVS, only the data falling within the distance you specify from the Hcell bounding box is processed.

If you are in an intermediate verification stage and need quick results, you can use FRAME BY after you run cell mode HLVS. Because Dracula does not process the entire cell, FRAME BY reduces the amount of data processed in composite mode.

For example, if you include this function in your rules file

```
HCELL INV INV1 FRAME BY 5
```

only the internal cell data falling within 5 microns of the Hcell bounding box is processed, as shown.



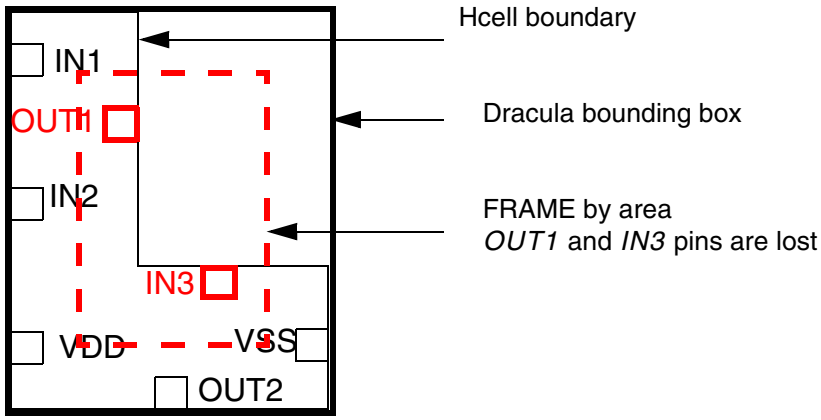
Caution

Do not use* `FRAME BY` *in the final verification stages because this function excludes portions of your design, masking potential problems.

Using Cell Boundaries

For each Hcell, Dracula creates a rectangular cell bounding box that surrounds all geometries in the Hcell. These boxes are put in a cell called `0CELLBOX` that you can overlay on your output cells. You can specify the layer number for the geometries in this cell with the `CELLBOX-LAYER` function in the Description block.

The following example shows a nonrectangular Hcell and the bounding box Dracula generates for it.



When you use the `HCELL FRAME BY` option or the `GEN-TEXT-FRAME` function for composite-mode HLVS, only data falling within the distance you specify from the Hcell bounding box is processed. For this example, if the Dracula-generated bounding box is used, two pins are lost, and the cell does not match the `.SUBCKT` I/Os in the netlist.

- To prevent this problem, you can create a graphic input layer with cell boundary geometries for nonrectangular cells. Specify this layer with the `CELLBNDY` function in the Input-layer block of your rules file.

  Dracula uses the `CELLBNDY` layer for the boundaries you create, and Dracula-generated bounding boxes for the Hcells for which you don't supply boundaries.

You can also use the `CELLBNDY` layer with `BASE-LAYER` for gate arrays. For example, if your cells consist only of metal but diffusion and poly are composite data, devices are electrically inside the cell but built from polygons outside the Dracula bounding box.

Marking Feedthroughs on the Layout

You might have interconnects (feedthroughs) on your layout that run through an Hcell without making connections to any devices in the Hcell. These feedthroughs do not have a counterpart in the netlist Hcells because they are part of the composite plane interconnect, not the Hcell. You must identify feedthroughs in your Hcell layout by giving them a text label of this form:

`FTHRU#`

where `#` is a number of your choice.

In cell mode, HVLS ignores feedthroughs. In composite mode, HLVS connects them to nodes.

➤ To generate feedthrough text automatically, use `GEN-TEXT-FLTNODE` in the Description block of your rules file.

See "*Using GEN-TEXT To Generate Hcell Text*" and "GEN-TEXT Commands" in the "Comparing Layouts and Schematics (LVS)" chapter of the *Dracula Reference.*

Using a Cross-Reference File for Hcell Names

For cell and composite mode HLVS and HLPE, Dracula requires layout cell names to match equivalent schematic `.SUBCKT` names. If you were not aware of this requirement when you created your schematics and layout, you can correct problems with cell names using an Hcell file (`HCELL-FILE = fileName`) to match names of Hcells between the schematic and layout.

Here are the contents of a sample `HCELL-FILE` that matches layout and schematic cell names:

```
NOR2  no2
NAND2 na2
REG   chip
```

Preparing Your Netlist

If HLVS is unable to complete the initial correspondence between layout Hcells and schematic Hcells, you might have to edit your netlist. Statements that you might have to include in your netlist are

- `*.EQUIV`
- `*.GLOBAL`
- `*.PIN`
- `*.NOPIN`

See the sections that follow for examples of how to use these statements.

For details, see "Compiling Network Descriptions (LOGLVS)" in the *Dracula Reference.*

Using a Netlist with Node Numbers

If your netlist is in true SPICE format, it uses node numbers rather than node names. To use this netlist in an HLVS run, you must include a local `*.EQUIV` statement for each Hcell to cross-reference the node numbers in the netlist to node names on the layout.

Here is an example:

```
.SUBCKT INV 3 17
*.EQUIV OUT=3 IN=17 VSS=0 VDD=99
M1 3 17 99 99 PMOS
M2 3 17  0  0 NMOS
.ENDS
```

Note: In composite mode, all Hcells must have corresponding `.SUBCKT` definitions in SPICE/CDL, but they need to define only I/Os, not transistors.

Adding Global Signals

You might have global Hcell I/O signals that are not defined as I/O signals in the `.SUBCKT` definition, but are required to match the layout Hcell with the netlist Hcell. To define these I/O signals in your SPICE/CDL netlist, use the `*.PIN` statement.

Any node defined by the `*.PIN` statement must be previously defined as a `*.GLOBAL` node. Here is an example:

```
*.GLOBAL VDD VSS
*.PIN VDD VSS
.SUBCKT INV OUT IN
M1 OUT IN VDD VDD PMOS
M2 OUT IN VSS VSS NMOS
.ENDS
```

If you put a `*.PIN` statement outside a `.SUBCKT` definition, the global signals you define apply to all Hcell `.SUBCKT`s. If you put the `*.PIN` within a `.SUBCKT` definition, the I/Os apply to only that subcircuit.

Removing Global Signals from Selected Subcircuits

For selected subcircuits, you can use the `*.NOPIN` statement to remove global I/Os that you previously defined in a `*.PIN` statement. Here is an example where the *CXFR* transfer gate does not need a connection to *VDD* on the layout to match the netlist.

```
*.GLOBAL VDD VSS
*.PIN VDD VSS
.SUBCKT INV OUT IN
M1 OUT IN VDD VDD PMOS
M2 OUT IN VSS VSS NMOS
.ENDS
.SUBCKT CXFR A B C
*.NOPIN VDD
M1 A B C VSS NMOS
.ENDS
```

Compiling Your Netlist

To compile your netlist, you must run LOGLVS just as you do for flat Dracula.

➤ To run HLVS, you must include the Hcell file name with a `CELL` command to LOGLVS when you compile your netlist.

For details and examples, see "Using Hierarchical LOGLVS" in the "Compiling Network Descriptions (LOGLVS)" chapter of the *Dracula Reference.*

Running Hierarchical Parameter Extractions

You use hierarchical Dracula in the same way you use flat Dracula when you extract parasitics from your circuit. To avoid duplicate processing, you extract parasitics from your cells prior to extracting the top-level design. Then, when you process the top-level design, cells are treated as blocks with pin names only.

For more information about how to run HLPE and HPRE, see these sections in the *Dracula Reference.*

- HLPE

  "Hierarchical LPE" in the "Extracting Electrical Parameters (LPE)" chapter

- HPRE

  "Hierarchical PRE" in the "Extracting Parasitic Resistance (PRE)" chapter

About Hierarchical Parameter Extraction

Initially, you might want to verify that your Dracula HLPE/HPRE rules file works on a flat database. You can then use it for hierarchical extraction, to take advantage of the smaller hierarchical netlist.

Extracting Capacitance Hierarchically

Hierarchical Dracula can extract capacitance between internal cell nodes in the same cell with the same placement during the cell mode run. However, nodes in one cell placement cannot be referenced from another cell. So although Dracula can extract internal capacitors, there is no way to describe them in a hierarchical netlist without making every node in every cell a pin on the netlist.

This problem appears in the following cases:

- Capacitance between internal cell nodes and nodes in the parent cell

  Dracula cannot list the capacitor in cross-coupled format because the node within the cell is not defined in the parent cell.

  In lump sum format, Dracula lists the capacitor between the node in the parent cell and ground. Dracula cannot list the capacitance between the internal cell node and ground as part of the parent cell.

  You can extract this capacitance by flattening layers.

- Capacitance between internal cell nodes of different cells

  There is no way to list this capacitor. Dracula can't list it for the parent cell of the two different cells, because neither node is in the parent cell. Dracula can't list it for the cell that contains the internal node unless the capacitor occurs in every placement. When Dracula extracts the cell capacitance, it can't determine how the cell will be placed in the future and what capacitors that placement will create.

- Capacitance between internal cell nodes of the same cell in different cell placements

  This case has the same problems as capacitance between internal cell nodes of different cells, described previously.

Flattening Layers To Extract Capacitance

You extract capacitors between internal cell nodes and parent cell nodes by flattening layers. To flatten layers, you flatten all nodes in the cell to the parent cell and assign all those nodes to ground. Here is an excerpt from a rules file that uses the flattening method:

```
;
; <functions omitted>
;
; Create flattened layer
FLATTEN METAL2 FLA2
;
; Set up nodal information for the flattened layer
LINK FLA2 TO VSS
;
; Create capacitance recognition layer
NOT METAL1 METAL2 MT1N2
AND MT1N2 FLA2 CA1F2 ; recognition layer for capacitance
                     ; between metal1 and fla2
;
; Extract parasitic capacitance
PARASITIC CAP[A] CA1F2 METAL1 FLA2 ; capacitors between
ATTRIBUTE CAP[A] 1 1               ; metal1 and fla2
;
; <functions omitted>
;
```

Do not build a recognition layer between a flattened layer and *BULK* if *BULK* is at the same potential, that is, the flattened layer and *BULK* have the same signal name.

The parent cell does not have to be the immediate parent. The parent cell and the cell containing the internal node can have several levels of hierarchy between them. You can use the flattening method for both cross-coupled and lump sum formats.

Note: If you list capacitors in cross-coupled format, the capacitors between an internal cell node and the parent cell are listed in lump sum format, which is the only way these capacitors can be listed in a netlist.

When you flatten layers, Dracula finds capacitors between nodes in both the child cell and the parent cell.

You can also use the `EXPLODE` function to extract capacitors between internal cell nodes and parent cell nodes.

Extracting Resistance Hierarchically

Unlike capacitors, resistors that have one node in an internal cell and one node in the parent cell are not really a problem. When you run extraction on

- The internal cell

  Dracula extracts the portion of the resistor that is in the cell.

- The parent cell

  Dracula extracts the resistance in the parent cell and connects the resistor terminal to the I/O of the cell. The pad layer defines where the I/O point of the cell is.

When you run HPRE to extract resistors hierarchically, each I/O point becomes a different subnode. If you have a signal that leaves the cell at more than one I/O point, check your output file for problems.

Writing Parameter Extraction Rules

The rules file you use for an HLPE or HPRE run is the same as the rules file for a flat run, with the addition of several hierarchy-related functions. The table in the *"Running a Hierarchical Network Comparison"* section lists the Dracula hierarchical functions that you might want to include in your rules file for an HLPE or HPRE run. For complete documentation about these functions, see the *Dracula Reference.*

Note: You cannot form devices by overlapping two Hcells. You can form elements in the Hcells or on the composite plane, but not between a combination of the two. The BASE-LAYER function is the only exception to this rule.

For a complete verification, you must run both cell mode and composite mode HLPE or HPRE.

Using a Cross-Reference File for Hcell Names

For cell and composite mode HLPE, Dracula requires that layout cell names and their equivalent schematic .SUBCKT names must be the same. If you were not aware of this requirement when you created your schematics and layout, you can correct problems with cell names by using an Hcell file (HCELL-FILE = *fileName*) to match the names of the Hcells in the schematic and layout.

For example, here are the contents of an HCELL-FILE that matches layout and schematic cell names:

```
NOR2  no2
NAND2 na2
REG   chip
```

Using HLPE To Extract a Complete Netlist

To use HLPE to extract a netlist of your complete circuit (Hcell plane and composite plane) in hierarchical Dracula, do the following.

1. Run a cell-mode HLPE.

   The transistor-level netlist contains .SUBCKT definitions for each Hcell listed in your HCELL-FILE. You specify the name of the netlist with the OUTPUT option of the LPESELECT function.

   ```
   LPESELECT[S] CAP GT 0.0 OUTPUT SPICE
   ```

   The cell-mode netlist has a .CEL extension.

2. Run a composite-mode HLPE, using the same OUTPUT file name with the LPESELECT function that you used for the cell-mode run.

   Dracula extracts a netlist with elements and parasitics from the composite plane and calls to the .SUBCKT definitions created in cell mode. This netlist is appended to the netlist extracted from the cell-mode run, making a complete two-level hierarchical netlist.

   The composite-mode netlist has a .DAT extension.

   Note: Dracula does not extract parasitics from composite-to-Hcell or Hcell-to-Hcell

overlaps.

Making Hierarchical Dracula Run Faster

Hierarchical Dracula is, in general, faster than flat Dracula on a hierarchical design, because it takes advantage of the repetitive data structures of that design. In this section, you will learn how to improve hierarchical Dracula performance by

- Maximizing hierarchical operations
- Minimizing redundant checking
- Altering Hcell selection criteria

For more help with improving hierarchical Dracula's performance, you can also see "*Improving Preformance*".

Maximizing Operations

To process a design, hierarchical Dracula must first set up the database by selecting Hcells, creating Hcell environments, and doing other hierarchical bookkeeping that flat Dracula does not do.

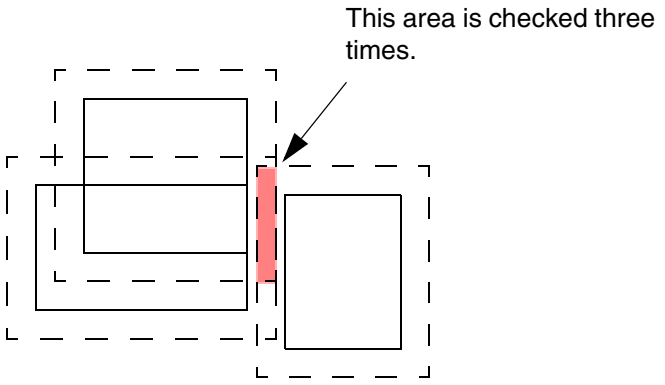
To make the best use of this setup time, include as many design rules as you can in the Operation section of your rules file. Because the setup time is the same whether you run a few rules or many, it is most efficient to run many rules at once.

When you are running hierarchical Dracula, you can improve processing time by removing checks that require nodal information and checks with large spacing values. Run these checks separately in flat mode.

Minimizing Redundant Checking

You can minimize redundant checking by avoiding Hcell overlaps. Dracula sets up Hcells by creating an environment area around them that is as wide as the largest spacing check on hierarchical layers in the Operations block. When Hcells overlap, their environments also

overlap. Because Dracula checks the environment of each Hcell, Dracula checks certain overlapping environment areas many times.



You can solve the problem of overlapping Hcells as follows:

- Use `ENVIRONMENT-MAX`

  Set this value as small as possible to minimize environment overlaps. To keep the environment value as small as possible, use flat checks for the operations with large spacing values. See *"Specifying a Maximum Environment Value"*.

- Use `EXCEPTION-ON=[HSP-OUTPUT]`

  Include this option in the Description block of your rules file so that Dracula does not select overlapping Hcells. See *"Preventing Hcell Overlap"*.

If your design contains a large number of overlapping cells, a flat run might take less time than a hierarchical run.

Altering the Hcell Selection Criteria

If you use Dracula's automatic Hcell selection and your job does not perform well, you may want to choose Hcells yourself. See *"Selecting Hcells Manually To Improve Performance"*.

You can alter the Hcell selection criteria with these functions:

- `HCELL`

  Specify names of cells you want to be Hcells. This list overrides automatic Hcell selection. To manually select Hcells, use `HCELL-FILE` to provide a list of selected Hcells in an external file.

- `NOT-HCELL`

Specify names of cells you do not want to be Hcells (HDRC only).

- `HCELL-RULE`

  Specify the minimum number of placements and segments for a cell to be an Hcell candidate. You can specify multiple placements/segments (HDRC only).

- `HCELL-MAX-SEGMENTS`

  Specify the maximum number of line segments for a cell to be an Hcell candidate, to avoid selecting large cells.

- `HCELL-HEIGHT-LIM`

  Set a limit of 1000 microns for the maximum Hcell height.

Index

Symbols

... in syntax 10
.CEL extension for cell-mode HLPE netlist 182
.DAT extension for composite-mode HLPE netlist 182
.erc file 171
.inp file 158, 171
.log file
    EXPAND stage and Hcell selection 158
    HDRC output 157
    HERC output 171
    reallocating memory 75
.mlg file 157, 171
.msm file 158, 171
.SUBCKT
    global signals 178
    in transistor-level netlist for HLPE 182
    names must match layout for HLVS/HLPE 177, 182
    pin names for HLVS/HLPE 65
    required for composite mode HLVS 173, 178
    used to generate text for LVS 68
.sum file 157, 171
*.EQUIV
    in SPICE netlists with node numbers 177
    preparing the netlist 177
*.GLOBAL
    add to HLVS netlist 178
    preparing the netlist 177
*.NOPIN
    preparing the netlist 177
    removing global signals from a subcircuit 178
*.PIN
    add to HLVS netlist 178
    preparing the netlist 177

Numerics

2.5D MB parasitic extraction 117
2D3B parasitic extraction 121

A

A option
    EXPLODE 97
    LPESELECT 96
accuracy
    2.5D MB parasitics 118
    constants 78
    electromigration checks 43
    fabrication and masking effects 78
allocating swap space 75
analysis, piecewise fringe capacitance 118, 121
AND
    grouping to improve performance 74
    passing nodal information 102
angles, flagging 48
antenna check 22
Applicon
    database conversion 21
    input format 20
area
    exact contact size check 35
    for 2.5D MB parasitics 117
    for sidewall capacitance with fringing effects 119
    in capacitance output format 85
    in EQUATION 88
    overlap capacitance 87
    parameter in PARSET 79
    ratio for antenna checks 22
    with ATTRIBUTE CAP 85
arrays
    check using RAM-CELL 153
    choosing Hcells 161
    environment 153
    gate. *See* gate arrays
    HDRC for 62
Assura RCX
    and the dracToRCX interface 135
    commands 140
ATTACH
    for Cadence-format input data 15
    used with GEN-TEXT-LAYER 69
    using 66
ATTRIBUTE CAP
    constants with 84
    debugging capacitance extraction 84
    definition 83
    examples
        colinear edge capacitance 92
        correcting for resistor cuts 114

directional sidewall capacitance 89
fringe capacitance in PRE 115
internal cell nodes to parent cell nodes 180
multiple for 2.5D MB extraction 119
simple capacitance 86
single-layer fringe capacitance 94
two-layer fringe capacitance 95
two-layer metal 107
piecewise analysis 118
replace with extractParasitic() 120
sidewall capacitance with fringing effects 118
specifying area and perimeter 85
values for single-layer fringe 95
ATTRIBUTE RES
definition 102
examples
contact 110
controlling contact cuts 117
diffusion 112
simple metal and poly 105
two-layer metal 107
replace with extractParasitic() 120

B

B option for SIZE 46
backannotation
using extractParasitic() 130
BASE-LAYER
definition 164, 173
for gate arrays 60
for HERC 167
in both cell and composite runs for a design 168
bipolar
recognition region for devices 49
rules file example 20
bounding box
created by CELLBOX-LAYER 163
displayed by SHORTBOX 170
for HLVS 175
generated by Dracula 176
specifying width with HCELL FRAME BY 174
using your own with CELLBNDY 176
BREAK
example 29
ungroups stages 75

C

C option 96
C parameter in PARSET 79
Cadence
  See also DFII 15
  output from Virtuoso 11
  using Cadence-format layout in Dracula 12
CALCULATE
  definition 23
  example 27
CANDIDAT.DAT 158
capacitance
  2.5D MB 117
  2D3B 121
  accuracy 78
  C parameter in PARSET 79
  colinear
    edge correction 90
    edge extraction 93, 121
  combined with resistance extraction 99, 120
  connecting power and ground 82
  cross-coupled
    definition 81
    hierarchical extraction limitation 180
    LPESELECT limitation 96
  debugging
    colinear edge correction 91
    fringe 84
    using constants 84
  directional sidewall
    definition 87
    illustration 88
    using extractParasitic() 121
  extraction functions 83, 120
  flattening to extract 180
  fringe
    CLL value in 2.5D MB 119
    debugging 84
    definition 80
    effects on sidewall capacitance 118
    extracting with extractParasitic() 121
    in 2.5D MB extraction 117
    in 2D3B extraction 121
    in PRE 114
    piecewise analysis 118, 121, 122
    single layer 94, 121
    single net 133
    TPR value in 2.5D MB 119
    two layers 95, 121

ground 82
lump sum
definition 81
in hierarchical extraction 180
internal cell nodes to parent cell nodes 181
LPESELECT C option omits 96
output as lump sum or cross-coupled 81
overlap
definition 80
example 85
extracting with extractParasitic() 121
rules file 85
two-layer metal 107
with fringing effects 118
over-the-cell wire 96
piecewise analysis 118, 121
power 82
sidewall
colinear edges 90
correcting after resistor cuts 112
directional 89, 121
simple 85, 121
single net 133
two-layer fringe 95
UNIT function 86
capgen 137
CASE
for Cadence-format input data 14
case sensitivity
Cadence-format input data 14
troubleshooting problems 14
CDLOUT
netlisting global signals 14
retains case information 14
cdsd 12
CDSIN
requirements 12
specifying the path to your libraries 12
using purpose data 15
when to use 11
CEL extension for cell-mode HLPE netlist 182
cell mode
description 63
for HLVS 174
HDRC error cells 154
products supported 146
text for 65
when to use 64, 71
CELLBNDY, using in composite mode 176
CELLBOX-LAYER
definition 147, 163, 172

using 175
CELL-CHILD-TEXT
definition 163, 172
using 67
CELL-ERROR-REP
ALL output 156
definition 147
generates Hcell errors 154
CHECK-MODE
cell 64, 154
composite 70
flat 58
for HDRC 147
for HERC 163
for HLPE 172
for HLVS 172
for HPRE 172
hierarchical
ALL error output 156
HIER error output 156
invoking 61
ONCE error output 155
viewing error cells 154
multilevel 63
checks
antenna rules 22
contact enclosure 32
corner 45
crosstalk 36
edge-related 45
electromigration 36
enclosure. *See* ENC
exact size 35
in-the-direction-of 31
metal reflection 36
spacing. *See* ENC; EXT; INT; WIDTH
CHKPAR
definition 23
examples
electromigration checks 38, 40, 42
using 23
clipping
current direction rules where allowed 33
current direction rules where not allowed 34
definition 32
CLL
calculation 119
definition 119
CMOS
extracting parasitic resistors 104
extracting two-layer metal parasitics 105

rules file examples 20
CNAMES-CSEN
for Cadence-format input data 14
colinear
edge capacitance extraction 93, 121
edge correction for parasitic extraction 90
command file. *See* rules file
commands. *See* functions
comments on output for InQuery 52
composite mode
creating a pad layer 101
description 70
for HLVS 174
frames in HLVS 174
HDRC error cells 157
HERC error cells 170
products supported 146
text for 72
when to use 71, 72
composite plane
definition 70
COMPUTE
criteria 23
definition 23
examples
antenna check 29
electromigration check 38, 40, 42
MAX option 25
MIN option 24
MINGATE option 26
SUM option 25
conjunctive rules
for edge checks 46
LENGTH in 46
CONNECT
connections not made 51
determining master layer for 50
does not require flat layers for HERC/HLVS 165
examples
antenna check 28
connecting power and ground 83
contact resistors 110
diffusion resistors 111
electromigration check 38
fringe capacitance in PRE 115
fuse extraction 43
over-the-cell routing wire capacitance to cell 97
for resistance extraction with extractParasitic() 130
in device extraction 50
intervening layers 51
requires master layer 50

when flat layers are required 153
CONNECT-LAYER
layers for contact resistance 130
master layer for contact layers 50
constants
for debugging capacitance 84
use by Dracula 78
with ATTRIBUTE CAP 84
contacts
as Hcells 161
basic enclosure check 32
connecting multiple layers 51
controlling resistor cuts 115
dimensions used in electromigration checks 36
divided 51
exact size check 35
extracting perimeter with EXT 41
extracting resistance 108, 127
in CONNECT functions 50
intervening layers 51
in-the-direction-of checks 31
master layer for 50
ratio of gate width to contact area 37
ratio of gate width to contact perimeter 41
ratio of gate width/length to contact area 38
ratio of gate width/number to contact area 40
conventions
rules file examples 10
syntax 10
user-defined arguments 10
coordinate system
in EDTEXT file 83
CORNER
example 45
to flag angles 48
corner
checks 45
markers 48
resistance 98
cross-coupled capacitance
between internal cell nodes and parent cell nodes 181
definition 81
hierarchical extraction limitation 180
LPESELECT limitation 96
crosstalk checks 36
CTEXT
definition 164, 173
layer for GEN-TEXT text 69
limitation 66, 74
using in cell mode 66
using in composite mode 73

current
drive 37, 38, 40, 41
in-the-direction-of checks 31
predicting peak 36
supplying values with CHKPAR 37
value, electromigration 37
CUT
definition 103
syntax 115
CUT-TERM
definition 102
ease of use 103
examples
contact 110
correcting capacitance for resistor cuts 113
fringe capacitance in PRE 115
poly 110
simple metal and poly 104
two-layer metal 106
extracts corner and junction resistance 98
limitations 115
MAXNS option 115
replaced by extractParasitic() 126
single-net extraction 134

D

D2RXRF 137
DAT extension for composite-mode HLPE netlist 182
database
See also layout 51
Dracula sets up for hierarchical processing 183
extracting a complete netlist in HLPE 182
format conversion 21
overlapping cells 183
size
large 60, 62
medium 58
small 58
x to y ratio 76
dbRead.cxt 12
ddbToRcx 138
debugging
capacitance
fringe 84
rules file 91
using constants 84
resistance 99
Design Framework II. *See* DFII
devices

bipolar 49
CMOS 20
extracting 48
NMOS 49
recognition region 48
DFII
See also Cadence 15
creating layout database 11
installation hierarchy 12
layer syntax in rules file 15
using layer purpose pairs 15
directional
DRC check in current direction 31
sidewall capacitance 87, 121
disk space
allocating with TRANSISTOR 76
allocating with TRANSISTOR-NUM 75
reduced requirements for composite mode 72
reduced requirements for hierarchical mode 62
distributed Dracula
description 60
when to use 60
dracToRCX 135, 138
modules 136
RCX commands 140
running the interface 141
Dracula
distributed 60
full functionality 58
hierarchical 145
Dracula To RCX Interface
See dracToRCX
DRAMs
checking 153
multilevel mode for 63
DRC
checks
antenna 22
corner 45
crosstalk 36
edge length 47
edge-related 45
electromigration 36
enclosure. *See* ENC
exact size 35
gate widths 47
in-the-direction-of 31
metal reflection 36
spacing. *See* ENC; EXT; INT; WIDTH
grouping checks to improve performance 74
hierarchical 146

multilevel 63
rules file example 20
drive current 37, 38, 40, 41
DSPF file
creating 103
rules file 105

E

edge
capacitance extraction, colinear 93
checks
converting partial edges to full edges 47
flagging angles 48
measuring edge length 47
using conjunctive rules 46
correction for colinear sidewall capacitance 90
EDTEXT
for composite mode 74
to correct case problems 14
ELCOUNT
in HERC composite run 165
results passed to Hcell pins 170
electromigration
checks
fuse extraction 42
gate approximation 36
definition 36
fuse extraction 42
ratio of gate width to contact area 37
ratio of gate width to contact perimeter 41
ratio of gate width/length to contact area 38
ratio of gate width/number to contact area 40
ELEMENT
example, MOS devices 49
single-net extraction 133
elements
resistors in HPRE 71
ENC
R and R' region options for edge checking 45
RC options to flag angles 48
RT options for measuring edge length 47
S square boundary option
basic check 32
description 32
in-the-direction-of check with no clipping 34
sizing R' results in edge checking 46
T option to convert partial edges to full edges 47
TR options to convert partial edges to full edges 47
when flat layers are required 152

enclosure checks. *See* ENC
environment
  and composite data 149
  automatically set 148
  definition 148
  for arrays 153
  for memories 153
  multiple overlaps 184
ENVIRONMENT-MAX
  definition 147
  minimizing redundant checking 184
  misses errors if too small 150
  using 149
EQUATION
  definition 79, 84, 103
  examples
    2.5D MB extraction 119
    colinear edge correction 93
    conjunctive rules 94
    contact resistance 110
    diffusion resistance 112
    extract colinear edge capacitance 93
  omit for default equation 120
EQUIV
  in SPICE netlists with node numbers 177
  preparing the netlist 177
ERC
  flat processing moved from HDRC 151
  hierarchical 163
  rules file examples 20
error cells
  composite
    HDRC 157
    HERC 170
  for CELL-ERROR-REP=ALL 156
  for CELL-ERROR-REP=HIER 156
  for CELL-ERROR-REP=ONCE 155
  Hcell 154
  HDRC 154
  HERC 168
  names 157
exact size checks 35
examples
  2.5D MB piecewise analysis 118
  2D3B capacitance 121
  antenna check 28
  colinear
    capacitance extraction 93, 121
    sidewall capacitance correction 90
  colinear edge capacitance 93, 121
  conjunctive rules 93

contact resistors 108, 127
controlling contact resistor cuts 115
corner check 45
correcting sidewall capacitance for resistor cuts 112
diffusion resistors 111
directional sidewall capacitance 87, 121
electromigration checks
fuse extraction 43
gate width to contact area 37
gate width to contact perimeter 41
gate width/length to contact area 39
gate width/number to contact area 40
exact size checks 35
flattening layers to extract capacitance 180
fringe capacitance in PRE 114
fuse extraction 43
gate width check 47
HCELL FRAME BY 175
in-the-direction-of checks 32
metal and poly resistors 104
overlap and sidewall capacitance 85, 121
over-the-cell routing overlap capacitance with cell geometries 96
pad check 152
poly resistors 126
region option with spacing checks 46
sheet resistance 126
sidewall capacitance with fringing effects 118, 121
single-layer fringe capacitance 94, 122
two-layer fringe capacitance 95, 122
two-layer metal parasitics 105
EXCEPTION-ON
definition 147
use for hierarchical SIZE 151
use to avoid redundant checking 184
use to disallow Hcell overlaps 150
with EXPLODE 98
EXPAND stage 158
EXPLODE
A option for merging cell and composite data 97
connections made with 50
example 97
extracting capacitance with 181
in rules file 98
EXT
can’t be used hierarchically 147
extract contact perimeter 41
pad check 152
R and R' region options for edge checking 45
RT options for measuring edge length 47
sizing R' results in edge checking 46
TR options in electromigration check 38

use to extract perimeter of contacts 41
when flat layers are required 152
extraction
capacitance 83, 120
colinear edge capacitance 93, 121
contact perimeter 41
devices 48
fuse network 42
hierarchical 180
layer purposes 15
nodal information 24, 38
resistance 102, 120
single net 133
text for HLVS 64
extractParasitic() 120
2D3B parasitics 121
colinear edge capacitance 121
contact resistance 127
description 120
directional sidewall capacitance 121
examples
four metal layers 132
two metal layers 131
layer definition 120
limitations 130
one-layer fringe capacitance 121
overlap capacitance 121
resistance 126
sheet resistance 126
sidewall capacitance 121
single-net extraction 134
syntax 128
two-layer fringe capacitance 121

F

fabrication effects 78
feedthroughs 176
files
.erc 171
.inp 158, 171
.log. *See* log file
.mlg 157, 171
.msm 158, 171
.sum 157, 171
DSPF. *See* DSPF file
EDTEXT. *See* EDTEXT
gen.rul 120
GEN-TEXT-FILE. *See* GEN-TEXT-FILE
Hcell list. *See* HCELL-FILE

HDRC output 154
HEDTEXT. *See* HEDTEXT
HERC output 168
InQuery 52
printed output. *See* PRINTFILE
rules. *See* rules file
SPICE. *See* SPICE
flagging angles 48
flat mode
description 58
for distributed processing 60
use for large number of overlapping cells 184
when to use 58
FLATTEN 101
definition 148, 164
examples
extracting capacitance 180
pad check 152
for large-value spacing checks 151
for pad layer 101
flattening
checks that require 152
EXPLODE 96
for HDRC nodal checks 165
generated pad layer 101
move from HDRC to HERC 151
not required for HERC/HLVS 165
to extract capacitance 180
FLATTEN-PWRGND
definition 164
using 166
format conversion 21
FRAME BY
and Hcell bounding box 176
and HERC 166
for composite-mode HLVS 174
may mask potential problems 175
not used in HERC 166
frames
for HLVS 174
using GEN-TEXT-FRAME 172
using HCELL FRAME BY 176
FRINGE CAP
definition 83
examples
interconnect in PRE 114
two-layer fringe 95
two-layer metal 107
using resistor terminals 114
R option extracts capacitance in resistor layers 114
replaced by extractParasitic() 120

fringe capacitance
2.5D MB considerations 117
CLL value in 2.5D MB 119
debugging 84
definition 80
effects on sidewall capacitance 118
extracting with extractParasitic() 120
in PRE 114
piecewise analysis 118, 121
single layer 94, 121
TPR value in 2.5D MB 119
two-layer 95, 121
FTHRU# 176
functions
AND. *See* AND
antenna checks 23
ATTACH. *See* ATTACH
ATTRIBUTE CAP. *See* ATTRIBUTE CAP
ATTRIBUTE RES. *See* ATTRIBUTE RES
BASE-LAYER. *See* BASE-LAYER
BREAK. *See* BREAK
CALCULATE. *See* CALCULATE
CASE. *See* CASE
CELLBNDY 176
CELLBOX-LAYER. *See* CELLBOX-LAYER
CELL-CHILD-TEXT. *See* CELL-CHILD-TEXT
CELL-ERROR-REP. *See* CELL-ERROR-REP
CHECK-MODE. *See* CHECK-MODE
CHKPAR. *See* CHKPAR
CNAMES-CSEN. *See* CNAMES-CSEN
COMPUTE. *See* COMPUTE
CONNECT. *See* CONNECT
CONNECT-LAYER 50
CORNER. *See* CORNER
CTEXT. *See* CTEXT
CUT. *See* CUT
CUT-TERM. *See* CUT-TERM
EDTEXT. *See* EDTEXT
ELCOUNT. *See* ELCOUNT
ELEMENT. *See* ELEMENT
ENC. *See* ENC
ENVIRONMENT-MAX. *See* ENVIRONMENT-MAX
EQUATION. *See* EQUATION
EXCEPTION-ON. *See* EXCEPTION-ON
EXPLODE. *See* EXPLODE
EXT. *See* EXT
extraction
capacitance 83, 120
resistance 101, 120
extractParasitic(). *See* extractParasitic()
FLATTEN. *See* FLATTEN

FLATTEN-PWRGND. *See* FLATTEN-PWRGND
FRINGE CAP. *See* FRINGE CAP
GEN-TEXT-FILE. *See* GEN-TEXT-FILE
GEN-TEXT-FLTNODE. *See* GEN-TEXT-FLTNODE
GEN-TEXT-FRAME 172
GEN-TEXT-LAYER. *See* GEN-TEXT-LAYER
GEN-TEXT-WIRE 172
grouping 74
HCELL. *See* HCELL
HCELL-COLUMN-1 172
HCELL-FILE. *See* HCELL-FILE
HCELL-HEIGHT-LIM. *See* HCELL-HEIGHT-LIM
HCELL-MAX-PLACEMENTS 148, 173
HCELL-MAX-SEGMENTS. *See* HCELL-MAX-SEGMENTS
HCELL-RULE. *See* HCELL-RULE
HEDTEXT. *See* HEDTEXT
HERC 163
HIERARCHEN. *See* HIERARCHEN
HLPE and HPRE 182
HLVS 172
INDISK. *See* INDISK
INDISK-FILE. *See* INDISK-FILE
KEEPDATA 52
LENGTH. *See* LENGTH
LEXTRACT. *See* LEXTRACT
LIBRARY. *See* LIBRARY
LINK 153
LPESELECT. *See* LPESELECT
LVSCHK. *See* LVSCHK
MODEL. *See* MODEL
MULTILAB. *See* MULTILAB
NEIGHBOR 152
NOT. *See* NOT
NOT-HCELL. *See* NOT-HCELL
PAD-LAYER. *See* PAD-LAYER
PARASITIC CAP. *See* PARASITIC CAP
PARASITIC DIO 44
PARASITIC RES. *See* PARASITIC RES
PARSET. *See* PARSET
PATHCHK. *See* PATHCHK
PLENGTH 47
PRINTFILE. *See* PRINTFILE
RAM-CELL 153
RCONNECT. *See* RCONNECT
RCONNECT-LAYER. *See* RCONNECT-LAYER
RELOCATE 147
requiring flat layers 152
RSPFSELECT. *See* RSPFSELECT
SCONNECT 50
SELECT. *See* SELECT
SIZE. *See* SIZE

SMART-LPE 133
SOFTCHK 165
STAMP. *See* STAMP
SUBNODE-DELIM 102
SYSTEM 17
TEXT. *See* TEXT
TNAMES-CSEN. *See* TNAMES-CSEN
TRANSISTOR 76
TRANSISTOR-NUM 75
UNIT. *See* UNIT
WIDTH. *See* WIDTH
fuse extraction 42

G

gate arrays
CELLBNDY with 176
limitations 60
using BASE-LAYER 167
gates
bent, measuring width of 47
finding width 38
parasitic resistor terminals 126
ratio of width to contact area 37
ratio of width to contact perimeter 41
ratio of width/length to contact area 38
ratio of width/number to contact area 40
sorting by number in series 40
sorting by size 39
GDSII
converting rules file to Cadence-format input 17
database conversion 21
input format 20
output from Virtuoso 11
GEN-TEXT
functions to extract Hcell text 65
using to create text file 68
using to text hierarchy 70
GEN-TEXT-FILE
definition 172
for HLVS cell mode 174
using 65
GEN-TEXT-FLTNODE
definition 172
using for feedthroughs 177
GEN-TEXT-FRAME 172
GEN-TEXT-LAYER
definition 173
example 69
using 69

GEN-TEXT-WIRE 172
GENXCN 136
GLOBAL
    add to HLVS netlist 178
    preparing the netlist 177
global signals
    adding to all subcircuits 178
    adding to SPICE/CDL netlist 178
    removing from SPICE/CDL subcircuits 178
ground
    flattening text from Hcell to composite level 166
    global signal in SPICE/CDL netlist 178
    included in extraction 96
    netlisted by CDLOUT 14
    required for capacitance extraction 82
    text requirement
        cell mode 65
        composite mode 72
grouping functions 74

H

HCELL
    and automatic Hcell selection 158
    definition 147, 164, 172
    for HLVS cell mode 174
    for HLVS composite mode 174
    FRAME BY option 176
    HLVS example 175
    selection criteria 158
    using to alter selection criteria 184
HCELL-COLUMN-1 172
HCELL-FILE
    definition 147, 164, 172
    to correct case of names 14
    to match schematic and layout names 177, 182
HCELL-HEIGHT-LIM
    definition 147, 172
    using to alter selection criteria 185
HCELL-MAX-PLACEMENTS 148, 173
HCELL-MAX-SEGMENTS 148
    definition 148, 173
    using to alter selection criteria 185
HCELL-RULE
    definition 148
    using to alter selection criteria 185
Hcells
    .SUBCKT requirement for HLVS 173
    altering selection criteria 184
    bounding boxes 175

bringing up child cell text 67
candidates 158
checking in cell mode 63
choosing to improve performance 160
correcting case of cell names 14
cross-referencing names between schematic and layout 177, 182
disallowing overlap 150
error cells 154, 156
extracting text for HLVS 64
feedthroughs 176
frames in HLVS 174
frames not used in HERC 166
global signals 178
improving performance 160, 184
list of candidates 158
minimizing redundant checking 183
multilevel criteria 63
multiple levels 63
no candidates identified 158
over-cell wire capacitance 96
overlapping 150, 183
selecting
for cell mode 64
for multilevel mode 63
to improve performance 160
text from external files
cell 67
composite 74
text from layout
cell 66
composite 73
text layer limitations 74
text requirements
cell mode 65
composite mode 72
using GEN-TEXT to supply text 68
verifying in cell mode 64
what is checked in hierarchical mode 61
HDRC
areas processed 61
automatic Hcell selection 158
ENVIRONMENT-MAX used in 150
error cells 154
flat checks in 151
generates Hcell error cells 154
grouping checks to improve performance 74
hierarchical mode 61
multilevel mode
for memories 153
using 63
output files 154

rules file
limitation 64, 71
writing 147
what is checked in composite mode 71
when flat layers are required 152
HEDTEXT
created by GEN-TEXT-FILE 172
definition 173
for composite mode 67
generated by GEN-TEXT 69
layer for GEN-TEXT 69
to correct case problems 14
HERC
error cells 168
output files 168
rules file 163
run modes 165
run with HLVS and HLPE 163
SHORTBOX cell 170
what is checked in composite mode 71
HIERARCHEN
definition 148
used in rules file 101
hierarchical mode
checks that can't be used 147
description 61
product supported 146
when to use 62
hierarchy
checks that can't be used 147
extracting a complete netlist 182
for HDRC 62
location of Hcells in 161
multiple levels 63
RC extraction limitations 179
texting with GEN-TEXT 70
using in Dracula 145
HLPE
extracting a netlist of the complete database 182
over-cell wire overlap capacitance 96
rules file 181
run HERC with 163
what is checked in composite mode 71
HLVS
extracting text in cell mode 64
netlist for 177
rules file
limitation 64, 71
writing 171
run HERC with 163
what is checked in composite mode 71

HPRE
composite mode for timing 72
resistance extraction limitations 181
rules file 181
what is checked in composite mode 71
htv command 52

I

INDISK
converting GDSII rules to Cadence 17
when to use 12
INDISK-FILE
contents 13
use to reference libraries 12
input formats
Applicon 20
Cadence 11
converting 21
GDSII 11, 20
InQuery 52
installation
hierarchy 12
instructions 9
instdir 12
INT
R and R' region options for edge checking 45
RT options for measuring edge length 47
when flat layers are required 153
interface, Dracula To RCX
See dracToRCX
in-the-direction-of checks 31
italics in syntax 10

J

junction resistance 98

K

KEEPDATA = INQUERY 52
keywords 10

L

L option for SIZE 46
LABEL, when flat layers are required 153

layers
associating comments with 52
Cadence syntax 15
DFII syntax 15
for MULTILAB check 169
for parasitic capacitance with extractParasitic() 120
grouping to improve performance 75
intervening, no connection made 51
master layer for CONNECT 50
purposes 15
using layer purpose pairs 15
layout
See also database 51
features that prevent connection 51
feedthroughs on 176
overlapping cells 183
Virtuoso output 11
x-to-y ratio problem 76
LENGTH
can't be used hierarchically 147
exact contact size check 35
input always partial edges in conjunctive checks 46
LEXTRACT
definition 79, 83, 102
examples
2.5D MB extraction 119
antenna checking 24, 25, 26, 27, 29
colinear edge capacitance 93
colinear edge correction 93
conjunctive rules 94
contact resistance 110
diffusion resistance 112
directional sidewall capacitance 89
electromigration check 38, 40, 42
omit for default equation 120
use in Flexible LPE 79
libraries
file requirements for Cadence input 12
license server requirements for Cadence input 12
multiple Cadence-format 13
specifying path 12
verified in cell mode 64
LIBRARY
converting GDSII rules file to Cadence 17
when to use 13
license server daemon 12
LINK, when flat layers are required 153
literal characters 10
log file
EXPAND stage for Hcell selection statistics 158
HDRC 157

HERC 171
reallocating memory 75
logical operations
checking for exact contact size 35
creating a pad layer 100
creating capacitor recognition layers 89
creating resistor recognition layers 103
examples
antenna checking 28, 30
capacitor recognition layers 92
colinear sidewall correction 91
create field poly 23
creating CMOS device layers 43
creating contact layers 42
creating parasitic diode layers 44
exact size checking 35
finding gates of different sizes 39
finding gates that are/are not in series 41
finding source/drain contacts 38
flat layers for capacitance extraction 180
pad check 152
extracting devices 49
extracting parasitic capacitance 83
passing node information 102
results in HDRC output cells 154
LOGLVS
CASE function for Cadence-format input data 14
htv command to generate InQuery files 52
using for HLVS 179
LPE
extracting parasitic capacitance 80
hierarchical 179
rules file examples 20
using external files 14
LPECHK
example 44
location in rules file 104
not needed in antenna check rules file 23
LPESELECT
A option to include power/ground nodes 96
C option to output cross-coupled capacitance 96
definition 84, 103
examples
2.5D MB extraction 119
contact resistance 110
contact resistors 131
controlling contact cuts 117
debugging colinear edge capacitance 93
extracting colinear edge capacitance 94, 131
extractParasitic() 131
fuse extraction 45

simple capacitance 86, 131
simple metal and poly resistors 105, 131
single-layer fringe capacitance 94, 131
two-layer fringe capacitance 95, 131
two-layer metal resistors 108
not needed in antenna check rules file 23
piecewise analysis 118
S option for schematic node names 105, 108
T option includes power/ground in report 96
to specify netlist name 182
lump-sum capacitances
between internal cell nodes and parent cell nodes 181
definition 81
in hierarchical extraction 180
LPESELECT C option omits 96
LVS
hierarchical 171
matching devices 49
rules file example 20
using external files 14
LVSCHK
example 86, 106
location in rules file 104
not needed in antenna check rules file 23

M

masking process effects 78
master layer for network connection 50
memories
checking using RAM-CELL 153
choosing Hcells 161
environment 153
HDRC for 62
multilevel HDRC for 161
metal reflection checks
avoiding false errors 46
example 36
metal resistors
four metal layers 132
one metal layer 104
two metal layers 105, 131
minimizing disk space
composite mode 72
hierarchical mode 61
MODEL
definition 83
modes
cell
Hcell error cells 154

HERC 163
HERC MULTILAB errors 168
HLPE 182
HLVS 174
HLVS cell names 177, 182
using BASE-LAYER 168
when to use 63
composite
HERC 163, 170
HLPE 182
HLVS 174, 176
HLVS cell names 177, 182
HLVS frames 174
using BASE-LAYER 168
when to use 70
distributed 60
flat
description 58
for distributed processing 60
use for large number of overlapping cells 184
when to use 58
for HDRC 146
for hierarchical products 146
hierarchical 148
automatic Hcell selection 158
error output 155, 156
FLATTEN 151
Hcell error cells 154
improving performance 160
when to use 61
multilevel
memories and arrays 153
when to use 62
modules for dracToRCX 136
MULTILAB
layers in output cell 168
SHORTBOX cell for 170
multilevel mode
description 62
for memories 153
when to use 63

N

NEIGHBOR in large spacing checks 152
netlist
compiling for HLVS 179
no feedthroughs in 176
preparing for HLVS 177
SPICE/CDL

for cell mode LVS/LPE 65
with node numbers 177
NMOS
device extraction 49
rules file examples 20
nodal information
checks and operations in flat mode 59
composite node numbers assigned by EXPLODE 96
extract for electromigration check 38
extract to use in antenna checking 24
for divided contact 51
for single node 133
fringe capacitance between different nodes 95
MULTILAB 168
node names in SPICE file 177
node numbers in netlist 177
passed by AND and NOT 102
requires flat layers 152
STAMP for antenna checking 28
texting for cell mode 66
transfer with AND 117
transfer with STAMP 113
use by HDRC and HERC 165
node names
attached to layers with GEN-TEXT-LAYER 69
changing SPICE node numbers to 177
global in SPICE/CDL 178
LVSCHK outputs schematic node names 86
NOPIN
preparing the netlist 177
removing global signals from a subcircuit 178
NOT
grouping to improve performance 74
passing nodal information 102
NOT-HCELL
definition 148
using to alter selection criteria 185

O

output
associating comments with 52
bounding box overlay 175
cell-mode netlist 182
composite-mode netlist 182
EXPLODE options 97
generating for capacitance extraction 83, 84
generating for resistance extraction 102, 103
HDRC 154
HEDTEXT file 68

HERC 168
LPESELECT options 96
of CUT function 115
schematic node names 86
selecting capacitance format 81
viewing shorts with SHORTBOX 170
overlap capacitance
definition 80
example 85, 121
rules file 85
two-layer metal 105
with fringing effects 118
overlap perimeter
definition 79
example 121
used in sidewall-down capacitance calculation 88
overlapping Hcells
disallowing overlap 150
minimizing redundant checking 183
over-the-cell wire overlap capacitance 96
OVPR
definition 79
examples
2.5D MB 119
colinear edge capacitance 93
colinear edge correction 91
conjunctive rules 94
correcting capacitance for resistor cuts 114
diffusion resistors 112
directional sidewall capacitance 89
in Flexible LPE 79

P

pad layer
flattening in HDRC 151
generating for resistance extraction 100
requirement for extracting resistance 99
PAD-LAYER
definition 102
for composite mode 101
terminates resistance 100
using 100
pad-to-metal check 151
parameters
examples
2.5D MB extraction 119
antenna checking 28
colinear edge capacitance 93
colinear edge correction 92

contact resistors 109
correcting capacitance for resistor cuts 113
diffusion resistors 111
directional sidewall capacitance 87, 89
electromigration 37, 39, 42
in Flexible LPE 79
parameter set 79
PARASITIC CAP
2.5D MB extraction 119
constants 78
definition 83
examples
colinear edge capacitance 92
correcting capacitance for resistor cuts 113
directional sidewall capacitance 89
internal cell nodes to parent cell nodes 180
simple capacitance 86
single-layer fringe capacitance 94
two-layer metal 107
piecewise analysis 118
replaced by extractParasitic() 120
PARASITIC DIO 44
parasitic extraction. *See* capacitance; resistance
PARASITIC RES
constants 78
definition 102
examples
contact 110
controlling contact cuts 117
diffusion 112
fuses 44
simple metal and poly 105
two-layer metal 107
replaced by extractParasitic() 120
PARSET
creating subtypes 131
definition 79
examples
2.5D MB extraction 119
antenna checking 23, 24, 25, 26, 27, 28
colinear capacitance 93
colinear edge correction 92
contact resistors 109
correcting capacitance for resistor cuts 113
diffusion resistors 111
electromigration 37, 39, 42
in Flexible LPE 79
not required with extractParasitic() 130
omit for default equation 120
PATHCHK
in HERC composite run 165

results passed to Hcell pins 170
performance
See also run times 160
improving by choosing Hcells 160
improving by grouping functions 74
PERI 79
perimeter
contacts 41
for sidewall capacitance with fringing effects 119
in capacitance output format 85
in EQUATION 88
in extractParasitic() 121
on one layer 79
overlapping a second layer 79
piecewise analysis 118, 121
PIN
add to HLVS netlist 178
preparing the netlist 177
pin purpose input 15
PLENGTH 47
poly resistors 104, 121
power
flattening text from Hcell to composite level 166
global signal in SPICE/CDL netlist 178
included in extraction 96
netlisted by CDLOUT 14
required for capacitance extraction 82
text requirement
cell mode 65
composite mode 72
PRE
cell mode limitation 63
extracting fringe capacitance 114
extracting parasitic resistance 98
hierarchical 179
rules file example 21
using external files 14
PRINTFILE
HDRC output files 158
HERC output files 171
purposes 15

R

R option
edge checking 45
flagging angles 48
measuring edge length 47
sizing results 46
R' option 46

RAM-CELL 153
ratios
    area sum, antenna 26
    field poly to gate 22
    gate width to contact area 37
    gate width to contact perimeter 41
    gate width/length to contact area 38
    gate width/number to contact area 40
    layer to devices on node 22
    layout x-to-y 76
    maximum area, antenna 25
    minimum area, antenna 24
    smaller of sum of areas, antenna 27
RCONNECT
    definition 102
    examples
        contact resistors 110
        controlling contact cuts 117
        correcting capacitance for resistor cuts 113
        diffusion 112
        simple metal and poly 104
        two-layer metal 106
RCONNECT-LAYER
    contact resistors illustration 108
    definition 102
    examples
        contact resistors 109
        diffusion 111
        fringe capacitance in PRE 115
        simple metal and poly 104
        two-layer metal 105
reducing disk space
    composite mode 72
    hierarchical mode 61
reflection rules
    avoiding false errors 46
    example 36
region options 45
RELOCATE 147
resistance
    2.5D MB 117
    accuracy 78
    contact 108, 127
    controlling contact cuts 115
    corner 98
    correcting sidewall capacitance for 112
    creating recognition layers 103
    debugging 99
    diffusion 111
    four-layer metal 132
    hierarchical extraction limitations 181

junction 98
pad layer required 99
poly 126
sheet 126
simple metal and poly 104, 131
single net 133
terminals used for fringe capacitance 114
two-layer metal 105, 131
validate capacitance before extracting 99
resolution unit
creating corner markers 48
criterion for edge length check 47
represents perimeter in electromigration check 41
used to convert edge portions to full edges 47
width of border in creating corner marker 48
width of error flags in region edge check 46
rotation
and RAM-CELL 153
to correct x-to-y ratio problem 76
RSPFSELECT
definition 84, 103
rules file
Cadence-format input 16
capacitance
2D3B 131
colinear sidewall 93
correct for resistor cuts 112
debugging colinear capacitance 91
directional sidewall 87, 131
fringe capacitance in PRE 114
overlap and sidewall 85, 131
sidewall capacitance with fringing effects 118, 131
single-layer fringe 94, 131
two-layer fringe 95, 131
two-layer metal 105
converting GDSII input to Cadence input 17
creating 20
database format conversion 21
DFII layer syntax 15
electromigration 37, 39, 40, 41, 43
fuse extraction 43
HDRC/HLVS limitation 64, 71
HLPE 181
HLVS 171, 173
HPRE 181
mixing flat and hierarchical commands 151
optimizing 76
output error cells in HDRC 154
parasitic extraction 131, 132
piecewise analysis 118, 132
resistance

contact 108, 131, 132
controlling contact cuts 115
diffusion 111
four-layer metal 132
metal and poly resistors 104, 131
two-layer metal 105, 131
single-net extraction 134
using BASE-LAYER 167
using TEXT and CTEXT 73
run modes
cell 63
composite 70
distributed 60
flat 58
hierarchical 61
multilevel 62
run times
HDRC limitations 62
reducing with black box LVS 72
reducing with distributed Dracula 60
reducing with Hcell selection 160
reducing with hierarchical mode 62

S

S option
ENC 32
LPESELECT 105, 108
SCONNECT 50
SELECT
grouping to improve performance 75
results in HDRC output cells 154
when flat layers are required 153
SHORTBOX cell 170
sidewall capacitance
2.5D MB considerations 117
colinear edge correction 90
sidewall-down 88, 121
sidewall-up 88, 121
with fringing effects 118, 121
signals
correcting case of signal names 14
global
adding to SPICE/CDL netlist 178
removing from subcircuits in netlist 178
single-net extraction 133
SIZE
B and L options for sizing regions 46
creating pad layer 100
creating pad layer in composite mode 101

for in-the-direction-of checks 34
for pad check 152
on the result of region operations 46
results in HDRC output cells 154
using for hierarchical sizing 151
sizing
for edge checks 46
for flattened pad layer check 152
for in-the-direction-of checks 34
hierarchical 151
to account for process effects 78
to create pad layer 100
SMART-LPE 133
SOFTCHK in HERC composite run 165
spacing checks. *See* ENC; EXT; INT; WIDTH
SPFSELECT
definition 84, 103
examples
simple metal and poly resistors 105
SPICE
changing node numbers to node names 177
for cell mode LVS/LPE 65
global signals in 178
STAMP
definition 102
examples
antenna checking 23, 26, 28
contact resistors 110
diffusion resistors 112
sidewall capacitance 113
simple metal and poly resistors 104
two-layer metal 106
when flat layers are required 153
subcircuits
global signals 178
in HLPE netlist 182
names must match layout for HLVS/HLPE 177, 182
pin names for HLVS/HLPE 65
removing global signals 178
required for composite mode HLVS 178
required for HLVS Hcells 173
used to generate text for LVS 68
SUBNODE-DELIM 102
swap space, preallocating 75
syntax conventions 10
SYSTEM function for Cadence rules file 17

T

T option

LPESELECT 96
spacing checks 47
TEXT
See also EDTEXT; HEDTEXT; text 74
for Cadence-format input data 15
limitation 66, 74
using 73
text
bringing up to Hcell level 67
cell mode requirements 65
excluded by FRAME BY 166
extracting from layout
cell 66
composite 73
flat mode requirements 60
from external files
cell 67
composite 74
generating with GEN-TEXT 68
Hcell layer limitations 74
hierarchy 70
in Cadence-format input data 16
in HDRC 165
in HERC/HLVS 166
requirements for composite mode 72
using cell mode to extract 64
TNAMES-CSEN
for Cadence-format input data 14
TPR
calculation 119
definition 119
TR options 38
TRANSISTOR 76
TRANSISTOR-NUM 75
transistors
estimate to preallocate swap space 75
not required in composite mode HLVS 178

U

UNIT
definition 83, 102
examples
antenna checking 28
single-layer fringe capacitance 94
two-layer fringe capacitance 95
output in SPICE file 86

V

Virtuoso 11

W

WIDTH
- exact contact size check 35
- R and R' region options for edge checking 45
- sizing R' results in edge checking 46
- using to create resistor recognition areas 117

width
- for exact size checks 35
- fringe capacitors 84
- gate
  - finding with EXT 38
  - in electromigration calculation 37

wire
- current flow of parasitic resistors on 108
- fringe capacitance 114
- overlap capacitance, over-the-cell 96
- over-the-cell overlap capacitance 96
- types automatically generated 172