



SIEMENS EDA

Calibre® xRC™ User's Manual

Software Version 2021.2
Document Revision 14

Unpublished work. © 2021 Siemens

This material contains trade secrets or otherwise confidential information owned by Siemens Industry Software, Inc., its subsidiaries or its affiliates (collectively, "Siemens"), or its licensors. Access to and use of this information is strictly limited as set forth in Customer's applicable agreement with Siemens. This material may not be copied, distributed, or otherwise disclosed outside of Customer's facilities without the express written permission of Siemens, and may not be used in any way not expressly authorized by Siemens.

This document is for information and instruction purposes. Siemens reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should, in all cases, consult Siemens to determine whether any changes have been made. Siemens disclaims all warranties with respect to this document including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement of intellectual property.

The terms and conditions governing the sale and licensing of Siemens products are set forth in written agreements between Siemens and its customers. Siemens' **End User License Agreement** may be viewed at: www.plm.automation.siemens.com/global/en/legal/online-terms/index.html.

No representation or other affirmation of fact contained in this publication shall be deemed to be a warranty or give rise to any liability of Siemens whatsoever.

TRADEMARKS: The trademarks, logos, and service marks ("Marks") used herein are the property of Siemens or other parties. No one is permitted to use these Marks without the prior written consent of Siemens or the owner of the Marks, as applicable. The use herein of third party Marks is not an attempt to indicate Siemens as a source of a product, but is intended to indicate a product from, or associated with, a particular third party. A list of Siemens' trademarks may be viewed at: www.plm.automation.siemens.com/global/en/legal/trademarks.html. The registered trademark Linux[®] is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

Support Center: support.sw.siemens.com

Send Feedback on Documentation: support.sw.siemens.com/doc_feedback_form

Revision History

Revision	Changes	Status/ Date
14	Modifications to improve the readability and comprehension of the content. Approved by Lucille Woo. All technical enhancements, changes, and fixes listed in the <i>Calibre Release Notes</i> for this products are reflected in this document. Approved by Michael Buehler.	Released April 2021
13	Modifications to improve the readability and comprehension of the content. Approved by Lucille Woo. All technical enhancements, changes, and fixes listed in the <i>Calibre Release Notes</i> for this products are reflected in this document. Approved by Michael Buehler.	Released January 2021
12	Modifications to improve the readability and comprehension of the content. Approved by Lucille Woo. All technical enhancements, changes, and fixes listed in the <i>Calibre Release Notes</i> for this products are reflected in this document. Approved by Michael Buehler.	Released October 2020
11	Modifications to improve the readability and comprehension of the content. Approved by Lucille Woo. All technical enhancements, changes, and fixes listed in the <i>Calibre Release Notes</i> for this products are reflected in this document. Approved by Michael Buehler.	Released July 2020

Author: In-house procedures and working practices require multiple authors for documents. All associated authors for each topic within this document are tracked within the Siemens EDA documentation source. For specific topic authors, contact the Siemens Digital Industries Software documentation department.

Revision History: Released documents maintain a revision history of up to four revisions. For earlier revision history, refer to earlier releases of documentation which are available on <https://support.sw.siemens.com/>.

Table of Contents

Revision History

Chapter 1

Calibre xRC Overview	15
The Calibre xRC Tool	15
Calibre xRC and the Physical Verification Flow	15
Syntax Conventions	17

Chapter 2

Getting Started: Parasitic Extraction Using Calibre Interactive	19
Invoking Calibre Interactive Parasitic Extraction (PEX)	19
Loading a Runset (Optional)	20
Specifying the Rule File for a PEX Run	20
Specifying Inputs	22
Defining Input Data Names in the Extracted Netlist	22
Using Schematic Netlist Input in the Extracted Netlist (Optional)	23
Defining H-Cells Input (Gate-Level, Hierarchical Extraction and ADMS Only)	23
Specifying Outputs for a PEX Run	25
Defining the Parasitic Netlist	25
Specify the Netlist	27
Restrict the Nets (Optional)	27
Set Up Reports (Optional)	28
Add to the SVDB (Optional)	29
Run Calibre Interactive PEX	29
Setting PEX Options	30

Chapter 3

Getting Started: Parasitic Extraction Using Calibre Batch Mode	33
Before You Begin	33
Step 1 — Create the PHDB	34
Step 2 — Create the Parasitic Database (PDB)	35
Step 3 — Output A Netlist or Report	38

Chapter 4

Types of Extraction	41
Comparison of Extraction Types	41
About the Example Design	42
Hybrid xACT 3D/Rule-Based Extraction	43
Hierarchical Memory Extraction	43
Mixed-Signal Hierarchical Extraction	46
In-Context Extraction	46
Gate-Level Extraction	47

Flat Transistor-Level Extraction	48
Chapter 5	
Producing Parasitic Models	49
Parasitic Devices	49
Types of Parasitic Models	50
Lumped Capacitance	50
Distributed Resistance	52
Distributed Resistance and Capacitance	52
Distributed Resistance and Coupled Capacitance	54
Chapter 6	
Basic Extraction Methods	57
Prerequisites for Performing Parasitic Extraction	58
Running Transistor-Level Extraction	59
Creating a Transistor-Level Netlist from the Command Line	59
Creating a Transistor-Level Netlist from Calibre Interactive	60
Running Gate-Level Extraction	62
Creating a Gate-Level Netlist from the Command Line	62
Creating a Gate-Level Netlist from Calibre Interactive	63
Running Full Hierarchical and Mixed-Signal Hierarchical Extraction	65
Creating a Hierarchical Netlist from the Command Line	65
Creating a Hierarchical Netlist from Calibre Interactive	66
Running ADMS Extraction	69
Creating an ADMS Netlist from Calibre Interactive	69
Extracting a Distributed RC PRIMETIME Netlist	76
Creating a PrimeTime Netlist from the Command Line	76
Creating a PrimeTime Netlist from Calibre Interactive	77
Extracting a Lumped C Spectre Netlist	79
Creating a Capacitance Netlist from the Command Line	79
Creating a Spectre Netlist from Calibre Interactive	80
Extracting a Netlist with Mixed Parasitic Networks	82
Mixing Parasitics from the Command Line	82
Mixing Parasitics from Calibre Interactive	83
Netlisting a Design Without Parasitics	86
Creating an Ideal Netlist from the Command Line	86
Creating an Ideal Netlist from Calibre Interactive	87
Backannotating Parasitics to a Source Netlist	89
Backannotating from the Command Line	89
Backannotating from Calibre Interactive	90
Generating a Capacitance Summary Report	91
Generating a Net-to-Net Coupling Capacitance Report	92
Reporting Coupled Capacitance from the Command Line	92
Reporting Coupled Capacitance from Calibre Interactive	93
Generating a Point-to-Point Resistance Report	95
Reporting Net Resistance from the Command Line	95
Reporting Net Resistance from Calibre Interactive	96
Extracting a Placed Cell	98

Table of Contents

Extracting Only the Top Level	99
Extracting a Block Using CB.	100
Running Calibre xRC CB from the Command Line.	100
Running Calibre xRC CB from Calibre Interactive	101
Chapter 7	
Handling Input	103
Hierarchy Control with Xcells.	104
Calibre nmLVS-H and Calibre xRC Cell List Compatibility.	105
Xcell List Format	105
Discovering Layout Paths for In-Context Cells	107
Wildcards in Xcell List.	108
Tips For Choosing Xcells for Full Hierarchical Extraction	109
Metal Fill Modeling.	109
Modeling Multiple Ground Regions	111
Varying Thickness with CMP Files.	112
Chapter 8	
Tuning Extraction.	115
Extracting Devices without Parasitics	115
Extracting Particular Nets	116
Wildcards and Search Level Specification	117
Excluding Power and Ground Nets	118
Grounding Coupled Capacitors	118
Ignoring or Extracting Floating Nets.	119
Extracting With Multiple Substrates	120
Resistance Extraction and PERC.	123
Chapter 9	
Controlling Netlisting.	125
Netlisting Multiple User-Defined Corners	125
Netlisting Only Direct Devices on a Selected Net.	126
Methods for Correcting Pin Swapping	127
Using the Source Based Flow.	128
How to Join a Disjoint Parasitic Model	130
Net Name Formatting	131
Verification of Timing with Probe Points	131
Parasitic Extraction with Calibre View Device Properties	131
Chapter 10	
Integration and Troubleshooting Topics	133
Integration	134
Guide to Calibre Interactive Files.	134
Creating Batch Shell Scripts Using Calibre Interactive	135
Best Practices for Shell Scripts.	137
Optimization	139
Improving Run Time	139
Creating Smaller Netlists	140

Best Way to Resize Designs	140
Troubleshooting	141
Setting Up For Troubleshooting	141
Invocation Issues	141
Chapter 11	
Handling Parasitic On-Chip Variation	143
On-Chip Variation in Parasitic Extraction	143
Parasitic Extraction Techniques for On-Chip Variation	146
In-Die Variation	146
Process Corners	147
CMP Modeling	148
Chapter 12	
Calibre xRC Tool Invocation Reference	151
Reference Syntax	151
Setting the CALIBRE_HOME Environment Variable	151
Command Invocation Reference	152
calibre -lvs	153
calibre -xrc -phdb	154
calibre -xrc -pdb	157
calibre -xrc -fmt	160
Appendix A	
Parasitic Effects and Calibre Tools	165
Parasitic Capacitors	165
Capacitance Models in Parasitic Extraction	166
Parasitic Resistors	169
Resistance Models in Parasitic Extraction	170
Appendix B	
Parasitic Extraction Commands	173
About SVRF Rules	173
Required Set	174
Required for Lumped Capacitance	177
Required for Distributed	178
Statements Specific To Distributed	179
Required for Source-Name Extraction	180
Example Pre-PEX SVRF File	181
Appendix C	
Output Reference	185
Databases	186
Parasitic Database (PDB)	186
Persistent Hierarchical Database (PHDB)	187
SVDB	188
Logfiles	189

Table of Contents

Netlists	190
Reports	193
Templates	195
Appendix D	
Reduction Techniques	199
Capacitive and Resistive Reduction	199
Threshold-based Reduction	200
TICER	201
Appendix E	
Time-it Tool Overview	203
Time-it Documentation	203
Time-it Application Overview	203
Appendix F	
Error and Warning Messages	207
Error Messages	207
Warning Messages	208
Glossary	
Index	
Third-Party Information	

List of Figures

Figure 1-1. Calibre xRC in the Physical Verification Flow	16
Figure 2-1. Loading Rules in Calibre Interactive	21
Figure 2-2. Providing Source Netlist to Calibre Interactive	23
Figure 2-3. Completing the H-Cells Tab	24
Figure 2-4. Extraction Mode Settings	25
Figure 2-5. Extraction Type Settings	26
Figure 2-6. Describing the Output Format	27
Figure 2-7. Enabling Report Output	28
Figure 3-1. Calibre xRC Extraction	33
Figure 4-1. SRAM Block Design Example	42
Figure 4-2. SRAM Block Design Hierarchy	43
Figure 4-3. Hierarchical vs. Actual RC Network Examples	45
Figure 5-1. Simplified Layout for Lumped Capacitance	50
Figure 5-2. Lumped Capacitance With Coupled Capacitor Added to Intrinsic Capacitors. .	51
Figure 5-3. Lumped Capacitance With Coupled Capacitor	51
Figure 5-4. Simplified Layout for Distributed Resistance	52
Figure 5-5. Distributed Resistance Extraction	52
Figure 5-6. Simplified Layout for Distributed Resistance and Capacitance	53
Figure 5-7. Distributed Resistance and Capacitance Extraction	54
Figure 5-8. Simplified Layout for Distributed Resistance With Coupled Capacitance.	54
Figure 5-9. Distributed Resistance With Coupled Capacitance	55
Figure 6-1. Extraction Mode Setting	60
Figure 6-2. Transistor Level Setting	61
Figure 6-3. Specifying HCell and XCell Files in Calibre Interactive	64
Figure 6-4. Extraction Mode	64
Figure 6-5. Gate Level Setting	64
Figure 6-6. Inputs Pane for Hierarchical Extraction	67
Figure 6-7. Set Extraction Mode	67
Figure 6-8. Outputs Pane for Hierarchical Extraction	67
Figure 6-9. Setup Verilog Translator	70
Figure 6-10. Setup Delay Calculation	71
Figure 6-11. Netlist Tab for ADMS	71
Figure 6-12. H-Cells Tab for ADMS	72
Figure 6-13. Explanation of Block Icons	72
Figure 6-14. Formatted Block	72
Figure 6-15. Changing Block Extraction Mode	73
Figure 6-16. Saving Blocks to Cell Files	73
Figure 6-17. Outputs Tab for ADMS	74
Figure 6-18. Transcript for Successful ADMS Run	74
Figure 6-19. xRC Extraction Mode Setting	77

Figure 6-20. R + C Mode Setting	78
Figure 6-21. PrimeTime Output Setting	78
Figure 6-22. Extracting a Lumped Capacitance Spectre Netlist	81
Figure 6-23. Enabling Extraction Steps in Calibre Interactive	84
Figure 6-24. Generating PDB Incrementally	84
Figure 6-25. Adding Nets to Existing PDB	85
Figure 6-26. Setting for No Parasitics	87
Figure 6-27. Coupling Capacitance Report Settings	93
Figure 6-28. Point-to-Point Resistance Report Settings	97
Figure 6-29. Enabling CB Runs in Calibre Interactive	102
Figure 7-1. Simple Metal Fill on a Single Layer	110
Figure 7-2. Metal Fill on Multiple Layers With Multiple Nets	111
Figure 7-3. Non-Square Fill With Multiple Nets	111
Figure 8-1. Device Extraction With and Without Parasitics	116
Figure 8-2. Floating-Net Coupling Example	119
Figure 8-3. Floating-Net Coupling Extraction Example	120
Figure 8-4. Parallel Line Example	122
Figure 9-1. Gate-Level Logical Pin Swapping Example	127
Figure 9-2. Comparison of Normal and Source Based Flows	129
Figure 11-1. Design Flow Showing In-Die	145
Figure 11-2. Layout Structure Affects Local Density	147
Figure 11-3. Typical Effects of CMP	149
Figure 11-4. Metal Fill and CMP	150
Figure A-1. Intrinsic Capacitance	167
Figure A-2. Coupled Capacitance	168
Figure C-1. Common Source/Drain Region	190
Figure C-2. Connection by Abutment	190
Figure C-3. General Template Structure	197
Figure E-1. Time-it Tool Design Flow	204
Figure E-2. Time-it Tool in the Design Flow	205

List of Tables

Table 1-1. Syntax Conventions	17
Table 2-1. Invocation Methods	20
Table 2-2. Tabs in Inputs Pane	22
Table 2-3. Tabs in Outputs Pane	25
Table 2-4. Adjustments Available in PEX Options	30
Table 3-1. Invocation Line for the PDB Step	38
Table 3-2. Formatter Options	40
Table 4-1. In-Context PDB Contents	46
Table 7-1. Xcell flags	106
Table 7-2. Wildcards in an Xcell List	109
Table 10-1. Calibre Interactive Settings Sequence	135
Table 10-2. Best Practices for Shell Scripts	137
Table F-1. Calibre xRC Error Messages	207
Table F-2. Calibre xRC Warning Messages	208

Chapter 1

Calibre xRC Overview

Calibre® xRC™ is a parasitic extraction tool that calculates parasitic resistance and capacitance in an IC layout, and generates parasitic netlists and reports.

The following sections provide an overview of the Calibre xRC product:

The Calibre xRC Tool	15
Calibre xRC and the Physical Verification Flow	15
Syntax Conventions	17

The Calibre xRC Tool

Calibre xRC is a parasitic extraction tool that generates parasitic netlists and reports. It calculates parasitic resistance and capacitance in an IC layout and outputs a simulatable netlist. In conjunction with the Calibre® xL product, it can also calculate parasitic inductance.

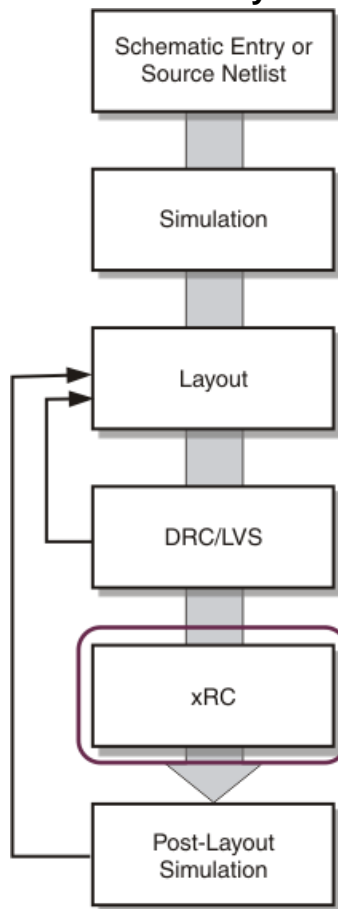
Use Calibre xRC to calculate the parasitics on your design. Parasitic effects can slow down signals, add noise, or cause hot spots in your design, among other problems.

Calibre xRC and the Physical Verification Flow

Run parasitic extraction after your layout passes LVS. Calibre xRC makes use of connectivity data. If the layout is not electrically correct, the parasitic results will not apply to the final design either.

After running parasitic extraction, you may choose to simulate your design. You can also use the reports to identify the most affected nets. Usually the cycle of physical verification, parasitic extraction, and post-layout verification is repeated several times before tapeout.

Figure 1-1. Calibre xRC in the Physical Verification Flow



Prerequisites for Calibre xRC

You must have the following to run the Calibre xRC tool:

- A layout database for which connectivity is defined.
- An optional source netlist if you require schematic names for the nets in the parasitic netlist.
- An SVRF rule file containing Calibre-specific SVRF statements and operations.
- The Calibre software.
- All necessary licenses. At a minimum, you must have a Calibre xRC, Calibre xRC to ADVance MS, or Calibre xRC CB license. Additional licenses may be required depending on the operations performed. For licensing information, see the “[Licensing: Parasitic Extraction Products](#)” section of the *Calibre Administrator’s Guide*.

How to Run Calibre xRC

You can run the Calibre xRC tool using either of following methods:

- Interactively from the Calibre® Interactive™ graphical user interface. See “[Getting Started: Parasitic Extraction Using Calibre Interactive](#)” on page 19.
- In batch mode from the command line. See “[Getting Started: Parasitic Extraction Using Calibre Batch Mode](#)” on page 33.

Syntax Conventions

The command descriptions use font properties and several metacharacters to document the command syntax.

Table 1-1. Syntax Conventions

Convention	Description
Bold	Bold fonts indicate a required item.
<i>Italic</i>	Italic fonts indicate a user-supplied argument.
Monospace	Monospace fonts indicate a shell command, line of code, or URL. A bold monospace font identifies text you enter.
<u>Underline</u>	Underlining indicates either the default argument or the default value of an argument.
UPPercase	For certain case-insensitive commands, uppercase indicates the minimum keyword characters. In most cases, you may omit the lowercase letters and abbreviate the keyword.
[]	Brackets enclose optional arguments. Do not include the brackets when entering the command unless they are quoted.
{ }	Braces enclose arguments to show grouping. Do not include the braces when entering the command unless they are quoted.
‘ ’	Quotes enclose metacharacters that are to be entered literally. Do not include single quotes when entering braces or brackets in a command.
or	Vertical bars indicate a choice between items. Do not include the bars when entering the command.
...	Three dots (an ellipsis) follows an argument or group of arguments that may appear more than once. Do not include the ellipsis when entering the command.

Table 1-1. Syntax Conventions (cont.)

Convention	Description
Example: DEVICE { <i>element_name</i> ['('model_name')']} <i>device_layer</i> { <i>pin_layer</i> ['('pin_name')'] ...} ['<'auxiliary_layer'>' ...] ['('swap_list')' ...] [BY NET BY SHAPE]	

Chapter 2

Getting Started: Parasitic Extraction Using Calibre Interactive

The Calibre xRC tool can be run from Calibre's Graphical User Interface (GUI) tool known as Calibre Interactive.

The following key sections in this chapter describe how to use Calibre Interactive to perform parasitic extraction:

Note



The steps in this chapter assume the Calibre software is already installed and licensing is properly set up.

Invoking Calibre Interactive Parasitic Extraction (PEX)	19
Loading a Runset (Optional)	20
Specifying the Rule File for a PEX Run	20
Specifying Inputs	22
Defining Input Data Names in the Extracted Netlist	22
Using Schematic Netlist Input in the Extracted Netlist (Optional)	23
Defining H-Cells Input (Gate-Level, Hierarchical Extraction and ADMS Only)	23
Specifying Outputs for a PEX Run	25
Defining the Parasitic Netlist	25
Specify the Netlist.	27
Restrict the Nets (Optional)	27
Set Up Reports (Optional)	28
Add to the SVDB (Optional)	29
Run Calibre Interactive PEX	29
Setting PEX Options	30

Invoking Calibre Interactive Parasitic Extraction (PEX)

The way in which you start Calibre Interactive depends on your tool environment.

Prerequisites

Environment correctly set up and configured:

- Calibre software installed and optionally integrated with layout editor.
- Calibre Interactive and Calibre xRC licenses available. See “[Licensing: Physical Verification Products](#)” and “[Licensing: Parasitic Extraction Products](#)” in the *Calibre Administrator’s Guide* for details.
- Either the \$MGC_HOME or \$CALIBRE_HOME environment variables defined. Calibre tools require that the CALIBRE_HOME environment variable be set. See “[Setting the CALIBRE_HOME Environment Variable](#)” in the *Calibre Administrator’s Guide* for details.

Procedure

Choose the appropriate method:

Table 2-1. Invocation Methods

From...	Use...
Cadence Virtuoso or other layout editor	Select Calibre > Run PEX from the menu
Calibre DESIGNrev	Select Verification > Run PEX from the menu
Command line	Type: calibre -gui -pex

Loading a Runset (Optional)

After you invoke Calibre Interactive PEX, you are prompted to specify a runset. A runset sets default values and can be useful for managing different types of extractions.

Procedure

1. To load a runset, use the file browser by clicking the Browse (...) button.
2. After navigating to the runset, select it and click **Open**. In the main dialog, click **OK**.
3. To skip loading a runset, click **Cancel**.

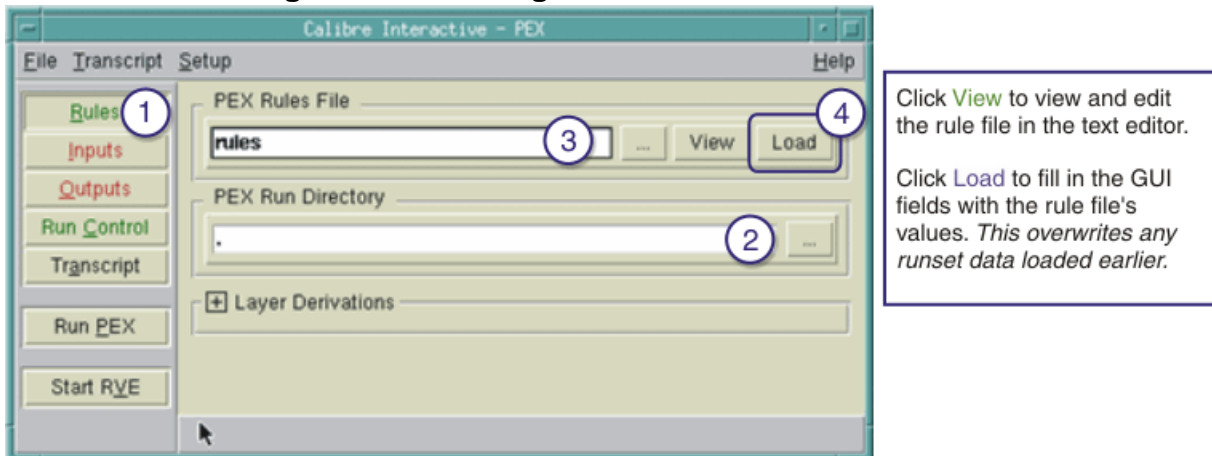
Specifying the Rule File for a PEX Run

This section describes the procedure used to specify a rule file for a PEX run.

Procedure

1. Click **Rules**.
2. Specify the run directory name. Use the Browse (...) button to select the run directory name from a list.
3. Specify the rule filename. Use the Browse button to select the rule filename from a list. Use the **View** button to view or edit the rule file.
4. Click **Load**. This loads GUI fields and sets GUI options based on rule file data.

Figure 2-1. Loading Rules in Calibre Interactive



Tip

- i** After you load a rule file, any information you specify in the GUI supersedes information in your loaded rule file.

Specifying Inputs

The source of your layout data varies depending on how you invoked Calibre Interactive and the type of extraction you plan to run.

Table 2-2. Tabs in Inputs Pane

Tab	Purpose	Required
Layout	Specify design database and format. (See “ Defining Input Data Names in the Extracted Netlist ”.)	Yes
Netlist	Specify <i>schematic</i> netlist or source files used when output should have schematic information. (See Using Schematic Netlist Input in the Extracted Netlist (Optional) .)	No
H-Cells	Specify cell lists used for performing a non-flat extraction. (See Defining H-Cells Input (Gate-Level, Hierarchical Extraction and ADMS Only) .)	No
Blocks	Specify blocks for xRC-ADMS extraction (See Running ADMS Extraction for information on ADMS extraction.)	No
Probes	Specify probe points.	No

Defining Input Data Names in the Extracted Netlist

This section describes the procedure used to specify the design database and format on the Layout tab.

Procedure

1. Click **Inputs**.
2. Choose **Layout**.
3. Specify the layout filename. The name appears in red text if the layout file does not yet exist.

If you invoked Calibre Interactive from a layout editor, you can use the current layout by selecting the Export from layout viewer option. (This will save a copy of the layout in the filename you specify. If the filename already exists, the contents will be overwritten.)

If there are multiple files for the layout:

- a. Enable the **PEX Options** pane by clicking **Setup > PEX Options**.
- b. Click the **Database** tab.

- c. Click the **Library** tab and use the **Add**, **Delete** and **Delete All** buttons to edit the list of additional layout files you would like to be available during your PEX run. Click **Inputs** to continue specifying input data.
4. Select a format type using the Format dropdown menu.
5. Specify the layout top cell name.

Using Schematic Netlist Input in the Extracted Netlist (Optional)

Follow this procedure when you want to use source names instead of layout names in the extracted netlist.

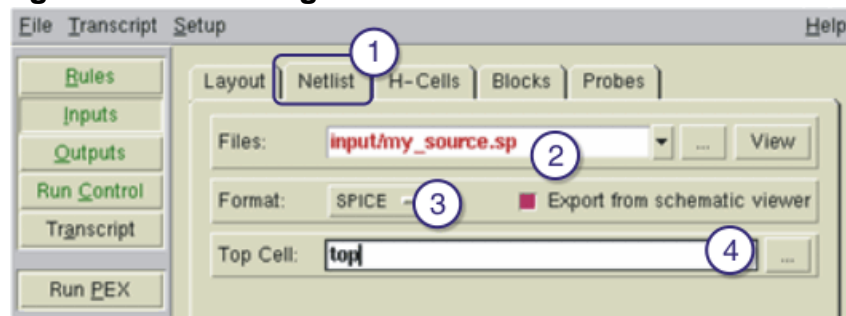
Procedure

1. Select the **Netlist** tab.
2. Specify the source netlist filename.

If you are also working with a schematic viewer, you can generate a new copy of a SPICE or Verilog netlist by choosing the Export from schematic viewer option. The schematic viewer must be running and the schematic data must be loaded in the schematic viewer's edit window.

3. Select the netlist file format.
4. Specify the source netlist top cell name.

Figure 2-2. Providing Source Netlist to Calibre Interactive



Defining H-Cells Input (Gate-Level, Hierarchical Extraction and ADMS Only)

Follow this procedure to specify cell lists when performing hierarchical extraction.

Note

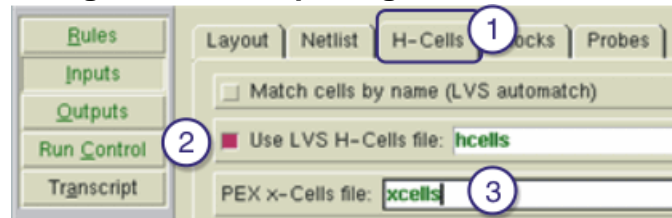


You do not need to perform these steps for flat extraction.

Procedure

1. Select the **H-Cells** tab.
2. Select Use LVS H-Cells file and specify the hcell filename (not required if you specify the use of layout names in the PEX netlist). The Calibre® nmLVS™ tool uses the hcell file. The Calibre xRC tool uses the xcell file. For more information on the xcell file, see [Hierarchy Control with Xcells](#).
3. Specify the xcell filename in the PEX x-Cells file field.

Figure 2-3. Completing the H-Cells Tab



Specifying Outputs for a PEX Run

Use the **Outputs** pane to specify the type of extraction.

Table 2-3. Tabs in Outputs Pane

Tab	Purpose	Required
(no tab)	Specify extraction mode and extraction type. (See Defining the Parasitic Netlist.)	Yes
Netlist	Specify the type of netlist. (See Specify the Netlist.)	Yes
Nets	Restrict the extraction to only some nets. (See Restrict the Nets (Optional).)	No
Reports	Set up LVS and xRC reports. (See Set Up Reports (Optional).)	No
SVDB	Add information types to the SVDB. (See Add to the SVDB (Optional).)	No

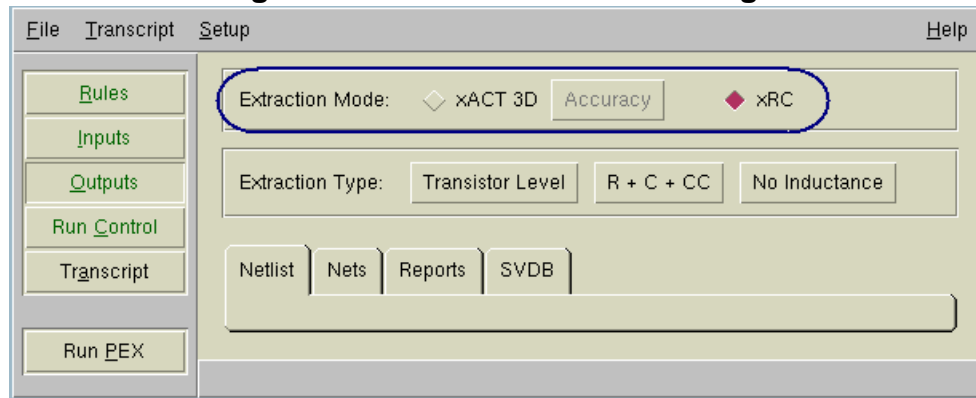
Defining the Parasitic Netlist

Use this procedure to specify extraction mode and extraction type.

Procedure

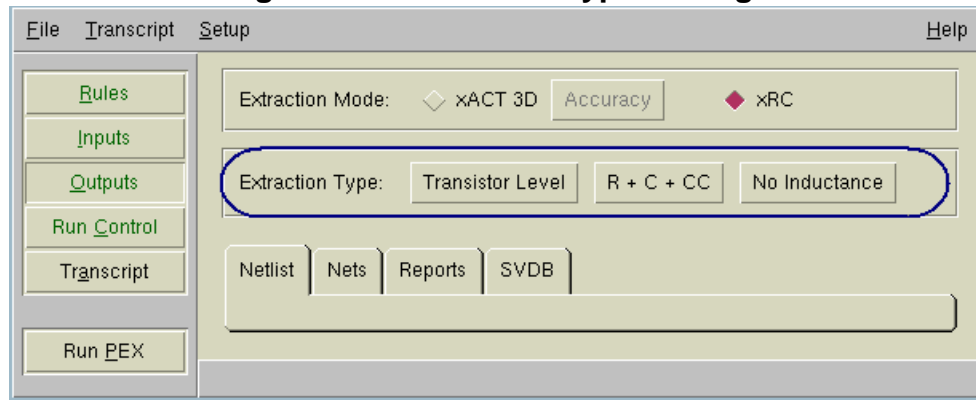
1. Click **Outputs**.
2. Choose the extraction mode xRC.

Figure 2-4. Extraction Mode Settings



3. Choose the extraction level (Transistor Level, Gate Level, Hierarchical, In-Context, or ADMS) from the first button on the Extraction Type: line.

Figure 2-5. Extraction Type Settings



Transistor Level is also known as “flat” extraction. Any cell placements are flattened into the top cell.

Gate Level extracts parasitics for geometries within the top cell, down to the boundary of the xcells. Xcells are specified in the file provided to the **Inputs > H-Cells** tab.

Hierarchical extracts parasitics for each identified xcell (not each cell placement) and the top cell. All geometries have parasitics extracted.

In-Context extracts parasitics for a particular cell in a full hierarchical extraction run. When this option is specified the -incontext command line option is added to the PDB step. See “[In-Context Extraction](#)” on page 46 for details.

ADMS extraction is similar to gate-level extraction. It provides additional automation for integrating analog and digital blocks for simulation and requires the Calibre xRC to ADVance MS license. See “[Licensing: Parasitic Extraction Products](#)” in the *Calibre Administrator’s Guide* for details.

See [Types of Extraction](#) for more details.

4. Choose the desired extraction type from the second button on the Extraction Type: line. Your choices are all combinations of R (resistance), C (intrinsic capacitance), and CC (coupled capacitance).

R + C also extracts coupled capacitance between nets but represents the value by adding it to the intrinsic capacitance. The combined intrinsic and coupled capacitance is also known as “lumped” capacitance.

5. Select the Inductance option to extract self-inductance and mutual-inductance parasitics. This option requires the Calibre xL license. See “[Licensing: Parasitic Extraction Products](#)” in the *Calibre Administrator’s Guide* for details. Inductance extraction is covered in detail in the [Calibre xL User’s Manual](#).

Note

The second button on the Extraction Type: line controls the information that is extracted into the PDB. The **Setup > PEX Options** dialog includes a “Parasitics to output to RC netlist:” field that controls the information from the PDB that is displayed in the netlist. In other words, you can set up your netlist to display a subset of what you have extracted to the PDB.

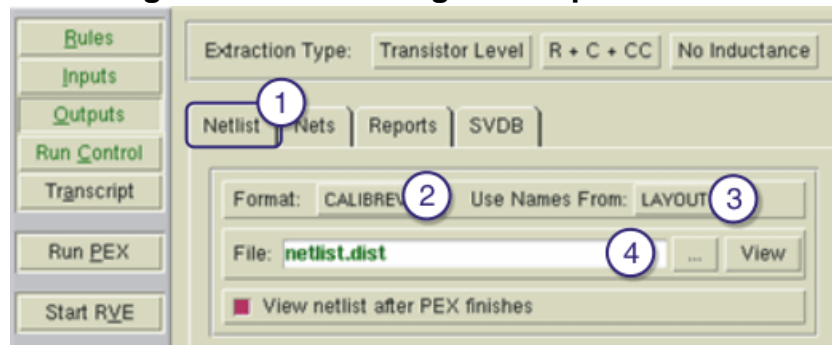
Specify the Netlist

Use this procedure to specify the netlist type you want to produce.

Procedure

1. Click the **Netlist** tab.
2. Choose the output format from the Format: dropdown list.
3. Choose the source (**SCHEMATIC** or **LAYOUT**) for pex netlist net and instance names from the Use Names From: dropdown list. If you choose SCHEMATIC, you must specify an LVS report name on the **Reports** tab.
4. Enter the PEX netlist filename.

Figure 2-6. Describing the Output Format



Restrict the Nets (Optional)

Use this procedure to restrict the extraction to only some nets. You can either exclude nets or use only the nets you specify. To decrease netlist size, it is useful to exclude power and ground nets unless they are central to the type of analysis you have planned.

Procedure

1. Click the **Nets** tab.
2. To exclude certain nets, select Specified Nets and then Exclude. Click the arrow button to browse the schematic viewer and select specific nets, or enter the netlist names

manually. The arrow buttons are active only if **SCHEMATIC** has been chosen in the **Netlist** tab.

Tip

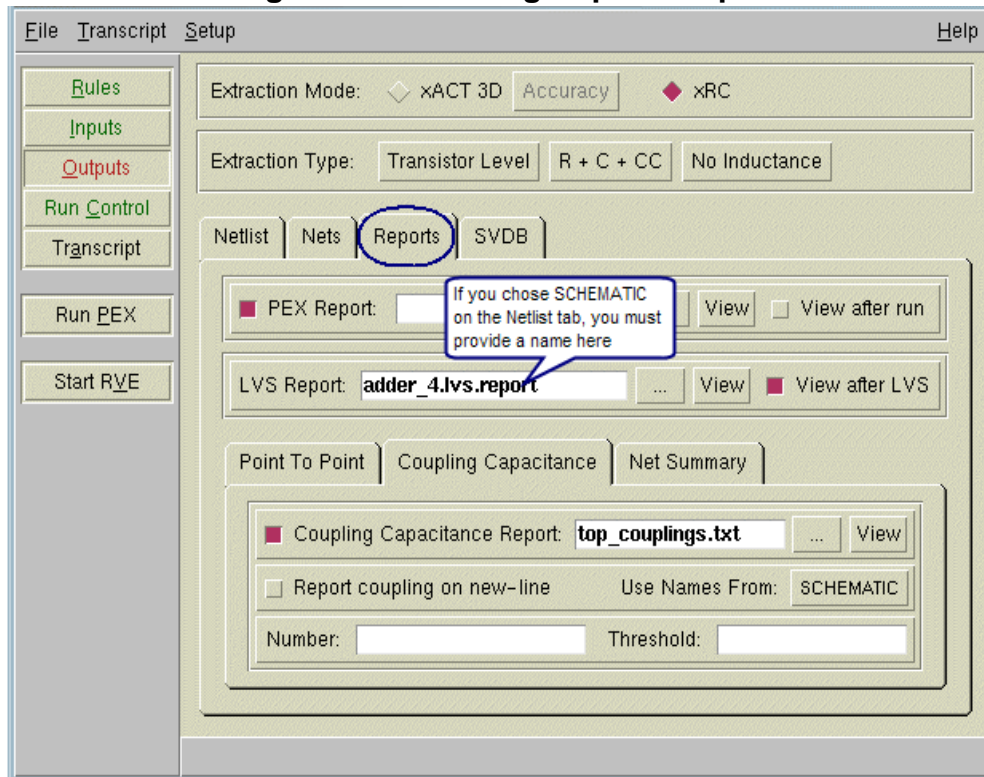
- i** Specify source net names if you chose the **SCHEMATIC** option for naming nets in the extracted netlist; specify layout net names if you chose **LAYOUT** naming.
-

Set Up Reports (Optional)

Use this procedure to set up LVS and xRC reports. There are five reports available.

- The PEX Report summarizes capacitance from the run.
- The LVS Report is the same as the one produced as part of a Calibre nmLVS run.
- The Point to Point Resistance report calculates resistance between two points on the same net. See [Reporting Net Resistance from Calibre Interactive](#).
- The Coupling Capacitance report summarizes capacitance between pairs of nets along with total capacitance of each net. This quickly shows the net pairs with the most significant coupling. See [Reporting Coupled Capacitance from Calibre Interactive](#).
- The Net Summary Report generates a report which details the parasitic capacitance values as stored in the PDB.

Figure 2-7. Enabling Report Output



Procedure

1. Click the **Reports** tab.
2. Choose report options and specify filenames as appropriate. You must specify an LVS report name if you chose SCHEMATIC from the Use Names From: dropdown list on the **Netlist** tab.

Add to the SVDB (Optional)

Use this procedure to add information types to the SVDB.

Procedure


1. Click the **SVDB** tab.
2. Specify the SVDB directory name.

The Generate cross-reference data for RVE option (enabled by default) generates the binary cross-reference database (XDB) for Calibre RVE and the query server.
3. If you need to start the Calibre RVE process on the selected SVDB file after PEX finishes, select the Start RVE after PEX option.
4. If you need to generate ASCII cross-reference (XDB) files in the SVDB database, select the Generate ASCII cross-reference files option.
5. If you need to generate Calibre Connectivity Interface (CCI) data in the SVDB database, select the Generate Calibre Connectivity Interface data option. This option requires a CCI license.
6. If you want to incrementally add to the PDB rather than re-generate the PDB every time the PDB generation step is run, select the Generate PDB incrementally.

Run Calibre Interactive PEX

After you have selected a runfile, and the inputs and outputs for your run, click the **Run PEX** button.

Tip

 All left panel buttons must be green before clicking **Run PEX**. A button changes from red to green when you have specified all required information associated with that button.

After the parasitic netlist is created, you can use Calibre RVE to highlight parasitic elements in the layout viewer. To run Calibre RVE, the SVDB must have RVE cross-references. This is set in the Outputs > SVDB tab, and is on by default. For detailed information on using the Calibre Results Viewing Environment (RVE) for PEX, see “[Using Calibre RVE for PEX](#)” in the *Calibre RVE User’s Manual*.

Setting PEX Options

You can override Calibre PEX defaults and customize the extracted parasitic netlist by selecting appropriate options which you access via the PEX Options panel.

Follow these steps to add the PEX options panel to the GUI.

Procedure

1. Choose the **PEX Options** option in the Setup menu.
2. Choose the **PEX Options** button in the left panel to display the options panel. Selecting the **PEX Options** item in the **Setup** menu adds the **PEX Options** button to the left panel button list.
3. Here are some of the capabilities:

Table 2-4. Adjustments Available in PEX Options

To...	Set...
	Netlist tab
Convert coupling capacitance into grounded capacitance. (The value of each grounded capacitor equals the value of the original coupling capacitor.)	On the Reduction and CC tab select Ground all coupling capacitors
Specify a name other than 0 for ground	On the Format tab select Ground node name
Create multiple ground regions using a layer	On the Format tab select Ground layer name
Specify the character that separates levels of hierarchy in the netlist (default is /)	On the Format tab select Hierarchy separator
Include additional parasitic resistor information in the netlist comments	On the Format tab select: Output intentional device ... Output values for intentional R,C devices Output parasitic R
Reduce RC and RCC netlists without affecting the time domain	On the Reduction and CC tab: Enable TICER reduction below
Reduce coupled capacitance parasitics	On the Reduction and CC tab: Enable CC reduction
Specify how floating nets such as metal fill are handled	On the Format tab: Extract floating nets
	Misc tab
Match the top-level pins to the order in the testbench	Create top level pin order

Table 2-4. Adjustments Available in PEX Options (cont.)

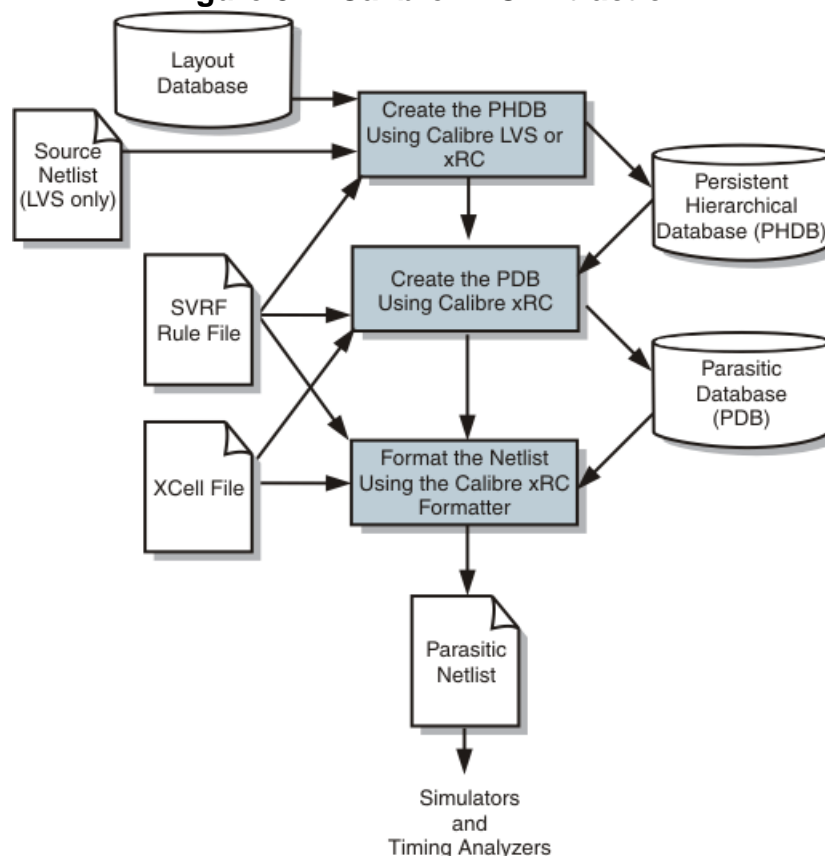
To...	Set...
Include only some parasitic types in the netlist	Parasitics to output to RC netlist
Generate a driver or receiver file	Generate driver/receiver

Chapter 3

Getting Started: Parasitic Extraction Using Calibre Batch Mode

There are three steps in the overall Calibre xRC extraction flow. Each step has many possible customizations based on your particular needs.

Figure 3-1. Calibre xRC Extraction



Before You Begin	33
Step 1 — Create the PHDB.....	34
Step 2 — Create the Parasitic Database (PDB).....	35
Step 3 — Output A Netlist or Report.....	38

Before You Begin

Running Calibre xRC extraction from the command line requires that certain conditions be met.

These instructions assume that everything is correctly set up and configured:

- Calibre software installed
- Calibre xRC license available
- Either the \$MGC_HOME or \$CALIBRE_HOME environment variables defined
- An SVRF file with layers, connectivity, devices, and parasitic calculations specified

See “[Setting the CALIBRE_HOME Environment Variable](#)” in the *Calibre Administrator’s Guide* for details.

Tip

 Shell scripts are an excellent way to run Calibre from the command line. A script can explicitly set environment variables and record invocation combinations you use frequently.

Step 1 — Create the PHDB

The Persistent Hierarchical Database, usually referred to as the PHDB, contains information about your design’s layout, connectivity, and devices necessary for calculating the parasitic information. You can create the PHDB with Calibre® nmLVS-H™ or Calibre xRC.

Procedure

1. Verify the SVRF file contains the following statements to specify the “inputs”, a layout database and source netlist.

- [Layout System](#), [Layout Path](#), and [Layout Primary](#) to specify the layout database.

The layout database, also called the design, is a physical representation of your IC. You identify this database in your SVRF rule file, or through the Calibre® Interactive™ Graphical User Interface. The Calibre software can read GDS, OASIS®¹, and LEF/DEF formats. LEF/DEF files may be specified in compressed (*.gz) or uncompressed format, if the compression utilities are in your environment.


- If you require source or schematic names in the generated netlist, use [Source System](#), [Source Path](#), and [Source Primary](#) to specify the source netlist.

If you require source (schematic) names in the generated netlist, you will need the source netlist. This netlist must be in SPICE format and is identified in the SVRF rule file or through Calibre Interactive.

2. Create the PHDB.

1. OASIS® is a registered trademark of Thomas Grebinski and licensed for use to SEMI®, San Jose. SEMI® is a registered trademark of Semiconductor Equipment and Materials International.

Note

 If your database is in LEF/DEF format, only layout names are supported. The PHDB must be created with `calibre -xrc -phdb`.

- **To use source names:**

```
calibre -lvs -hier -spice directory_path/filename.sp SVRF_file
```

For the output to be used in extraction, *directory_path* must match that specified in the Mask SVDB Directory statement, and *filename* must be the cell name of the Layout Primary statement. For details on invocation options for PHDB generation with Calibre nmLVS, see “[calibre -lvs](#)”.

- **To use layout names:**

```
calibre -xrc -phdb SVRF_file
```

The PHDB is only generated once per layout. You do not need to regenerate the PHDB unless your design changes, or you modify the SVRF connectivity rules. For details on invocation options for PHDB generation with Calibre xRC, see “[calibre -xrc -phdb](#)”.

Results

Calibre notifies you when it has successfully completed. Check the transcript for any errors.

After creating the PHDB, these files are created:

- **PHDB** — The PHDB is stored in the Standard Verification Database (SVDB) and used in creating the PDB.
- **LVS Report file (default: *lvs.rep*)** — If you used Calibre® nmLVS™, it will write the results of the LVS run to this ASCII file. Review this report to verify that there are no LVS or connectivity extraction problems.
- **LVS Extraction file (default: *lvs.rep.ext*)** — If you used Calibre nmLVS, it will write the results of the circuit extraction to this file. Review this report to verify that there are no LVS or connectivity extraction problems.
- **Layout netlist** — This is the SPICE netlist that the Calibre xRC tool will use for input in the next step. This file is named after the top level cell.

Related Topics

[Persistent Hierarchical Database \(PHDB\)](#)

Step 2 — Create the Parasitic Database (PDB)

Once you have created the PHDB, you create the PDB. The PDB stores the parasitic models for each extracted net.

This step can be run multiple times without regenerating the PHDB. You might want to do this if you are extracting different types of parasitics on different nets or if you are also extracting inductance with the Calibre® xL Parasitic Inductance Engine. For information on extracting parasitic inductance, see the [Calibre xL User's Manual](#).

There are many decisions that affect how you create the PDB. The most important is speed versus level of detail, which affects accuracy. Parasitic extraction for an entire chip can take from hours to more than a day. (The exact duration depends on the capabilities of the computer on which you run the analysis and the number of nets in the IC design.)

Prerequisites

- **PHDB** created in [Step 1 — Create the PHDB](#).

If you are doing multiple runs, you must use the same PHDB each time. The PHDB must be in the SVDB directory; you cannot separately specify the location.

- **Xcell file** (non-flat extraction only).

When performing hierarchical extraction, the hierarchy is specified by means of an xcell file. If you are doing a transistor-level extraction, or working with LEF/DEF layouts, you do not need an xcell file.

- **SVRF rule file.**

This should be the same file as used in [Step 1 — Create the PHDB](#).

Procedure

1. Determine which parasitics you need. The choices are

- resistance (-r)
- lumped capacitance (-c)
- resistance and distributed capacitance (-rc)
- resistance with distributed capacitance and coupled capacitance between nets (-rcc)

There is a trade-off between the amount of detail and how long the netlist takes to simulate. For example, a netlist with parasitics as lumped capacitance takes less time to simulate than one with coupled capacitance between nets, which takes less time than one with coupled capacitance between nets including floating nets.

2. Determine how much of the design you need to extract.

The less of the design you perform parasitic extraction for, the faster simulation will run. Your choices are

- “Flat” transistor-level extraction, which flattens all design hierarchy and extracts parasitics for everything not explicitly excluded. This is the default if no xcell list is added. (If you are using Calibre xRC-CB, you must run flat extraction.)

- Gate-level extraction, which ignores portions of the design listed in an xcell file. (The portions are usually devices or standard cells that have been verified separately.) The part of the design being extracted is flattened, just as with transistor-level extraction.
- Full hierarchical extraction, which extracts each cell listed in an xcell file only once. This method is only recommended for designs with large, symmetrically placed cells such as memory chips.
- Any of the above, with “select net”. This option extracts only nets explicitly specified. Use this with iterative extraction to handle critical nets: first most of the design is extracted with minimal detail, and then extraction is run again selecting the critical nets with more parasitic detail.
- Special case extraction methods, such as ADMS and in-context cells.

Note

If you are performing multiple extractions on the same design into a single PDB, the extractions need to all be the same type (for example, all gate-level).

Transistor-level (“flat”) extraction is the most accurate, but also slow. For large designs, flat extraction may produce netlists that are too large to simulate. However, where accuracy is critical, transistor-level extraction on a limited section of the design provides detailed information.

- a. If you decide on gate-level or full hierarchical extraction, construct an xcell list. See [“Xcell List Format”](#) for more explanation.
 - b. If you decide on select-net extraction, add a [PEX Extract Include](#) SVRF statement to your SVRF file.
3. Run the extraction.

The PDB step always contains at least the following:

```
calibre -xrc -pdb parasitic_switch SVRF_file
```

where *parasitic_switch* is determined in Step 1; for example,

```
calibre -xrc -pdb -rc rules.svrf
```

The above example performs a transistor-level extraction for resistance and distributed capacitance using the settings specified in the file *rules.svrf*, and any files it included. More simple examples are shown in the [“Basic Extraction Methods”](#) chapter.

4. If you need additional detail on parts of the design (for example, getting precise couplings for critical nets), run extraction again using the same files.

For example,

```
calibre -xrc -pdb -rcclm -select rules.svrf
```

[Table 3-1](#) shows common options for the PDB creation step. The decisions you made earlier let you choose among the extraction and parasitic options. Only one extraction option and one parasitic option can be specified per run.

Table 3-1. Invocation Line for the PDB Step

Required	Extraction Options ¹	Parasitic Options ¹	Additional Options	Required
calibre -xrc -pdb	no switch (flat)	-c	-cb	<i>rule_file</i>
	-xcell <i>xcell_file</i>	-r	-select / -cselect	
	-xcell <i>xcell_file</i> -full	-rc		
	-xcell <i>xcell_file</i> -incontext	-rcc -adms		

1. Choose only one from this column.

The procedures in “[Basic Extraction Methods](#)” give complete command line invocations. For details on invocation options, see “[calibre -xrc -pdb](#)”.

Results

Calibre xRC ends the transcript with a summary of errors and warnings. Be sure to check for any errors; these invalidate results.

This step creates the PDB directory in the SVDB directory. The PDB stores the parasitic models for each extracted net. These models consist of the net’s name and the collection of device pins, ports, parasitic delays, and circuit elements.

Related Topics

[Types of Extraction](#)

[Producing Parasitic Models](#)

[Hierarchy Control with Xcells](#)

Step 3 — Output A Netlist or Report

As the last step, you produce a netlist or report using the Calibre xRC formatter. The netlist can be in any of several formats such as HSPICE or DSPF. You can also set the formatter to perform different types of reductions to produce netlists that are more easily simulated.

Prerequisites

- **PHDB** created in [Step 1 — Create the PHDB](#).
The PHDB contains connectivity information.
- **PDB** created in [Step 2 — Create the Parasitic Database \(PDB\)](#).

The PDB contains the parasitic information.

- **SVRF rule file.**

This should be the same file as used in Steps 1 and 2.

Procedure

1. Determine the output you need.

- Reports are useful for identifying the nets or cells most affected by parasitics, or for post-processing in spreadsheets or with scripts. Reports can be generated at the same time as netlists.
- Netlists are useful for simulation. The format (SPICE, DSPF, CalibreView...) you need depends on your simulator. Only one format of netlist is generated at a time.

2. Verify the SVRF file contains the necessary statements for your output.

- Reports — Any of the following:

Coupled Capacitance	PEX Report Coupling Capacitance
---------------------	---

Total Capacitance	PEX Report Netsummary
-------------------	---------------------------------------

Resistance	PEX Report Point2Point
------------	--

- Netlist — The format is specified in a PEX Netlist statement, depending on the parasitics that were extracted.

Lumped or Distributed	PEX Netlist
-----------------------	-----------------------------

No Parasitics	PEX Netlist Simple
---------------	------------------------------------

ADMS	PEX Netlist ADMS
------	----------------------------------

3. Output the netlist or report(s).

This step always contains at least the following:

```
calibre -xrc -fmt SVRF_file
```

The output depends on what values are set in the SVRF file and what parasitics were extracted in [Step 2 — Create the Parasitic Database \(PDB\)](#).

For example, if the PDB contains RC values and the SVRF specifies “PEX Netlist design.spf SPEF PRIMETIME”, Calibre xRC writes out a SPEF netlist suitable for the PrimeTime® static timing analysis program containing distributed resistance and capacitance parasitic models. Any specified reports are also created.

The formatter provides the following additional options for specifying an output netlist:

Table 3-2. Formatter Options

Option	Result
-c	Writes only capacitance models into netlist, even if PDB contains more information.
-r	Writes only resistance models into the netlist, even if PDB contains more information.
-rc	Writes distributed resistance and capacitance models into the netlist.
-rcc	Writes distributed resistance and coupled capacitance models into the netlist.
-all ¹	Writes distributed results; does not produce lumped capacitance.
-corner	Writes netlists for specified <i>process corners</i> defined in the rule file.
-adms	Produces netlists suitable for Siemens EDA ADVance/MS mixed-signal simulator.
-simple	Writes only the intentional netlist, without parasitics. Useful for verifying that the design is being interpreted correctly.

1. This is the default, and recommended for most netlists.

Results

Calibre xRC ends the transcript with a summary of errors and warnings. Be sure to check for any errors; these invalidate results.

The working directory also contains the requested netlist and reports.

Related Topics

[calibre -xrc -fmt](#)

[Controlling Netlisting](#)

[Netlists](#)

Chapter 4

Types of Extraction

The Calibre xRC extraction engine is able to extract interconnect parasitics hierarchically. Hierarchical extraction is usually faster and requires less memory; the hierarchical netlists it creates are also smaller than equivalent non-hierarchical netlists and easier to simulate. This chapter provides additional information on each of the types of extraction and reviews their trade-offs.

Comparison of Extraction Types	41
About the Example Design	42
Hybrid xACT 3D/Rule-Based Extraction	43
Hierarchical Memory Extraction.....	43
Mixed-Signal Hierarchical Extraction.....	46
In-Context Extraction	46
Gate-Level Extraction	47
Flat Transistor-Level Extraction	48

Comparison of Extraction Types

A design can be extracted in several different ways. Which one is right for you depends on your design stage and what you intend to do with the netlist.

Depending on your design stage, different extraction types may be more accurate. For example, flat extraction preserves all the interaction between components, but if you are doing a mixed signal design then mixed-signal hierarchical extraction with its primitives allows you to enter parameters for a pre-characterized device. Similarly, if some parts of the design are to be supplied later, a flat extraction provides misleading results compared to a gate-level extraction with xcells for the missing parts.

The other consideration is the size of the produced data and netlist. Large designs may require several gigabytes of memory for the PHDB and PDB. Flat extraction requires more space than any form of hierarchical extraction. If both completeness of certain regions and size of output are important, use iterative runs with mixed-signal hierarchical extraction to get compact, detailed models on some areas, and a reference that you can fill in with a simplified model in others.

About the Example Design

The example design is a simple SRAM memory block which uses design hierarchy. The layout shows some of the block-level routing and major sub-circuits in the design, representing a simple design and how routing interacts with the different sub-circuits.

Figure 4-1 shows the layout used to illustrate the output of the extraction types. There are more cells and intentional devices at each level. For example, the *word* block is composed of *bit* cells. Each *bit* cell is composed of NMOS and PMOS transistors. These details are not shown in the layout.

Figure 4-1. SRAM Block Design Example

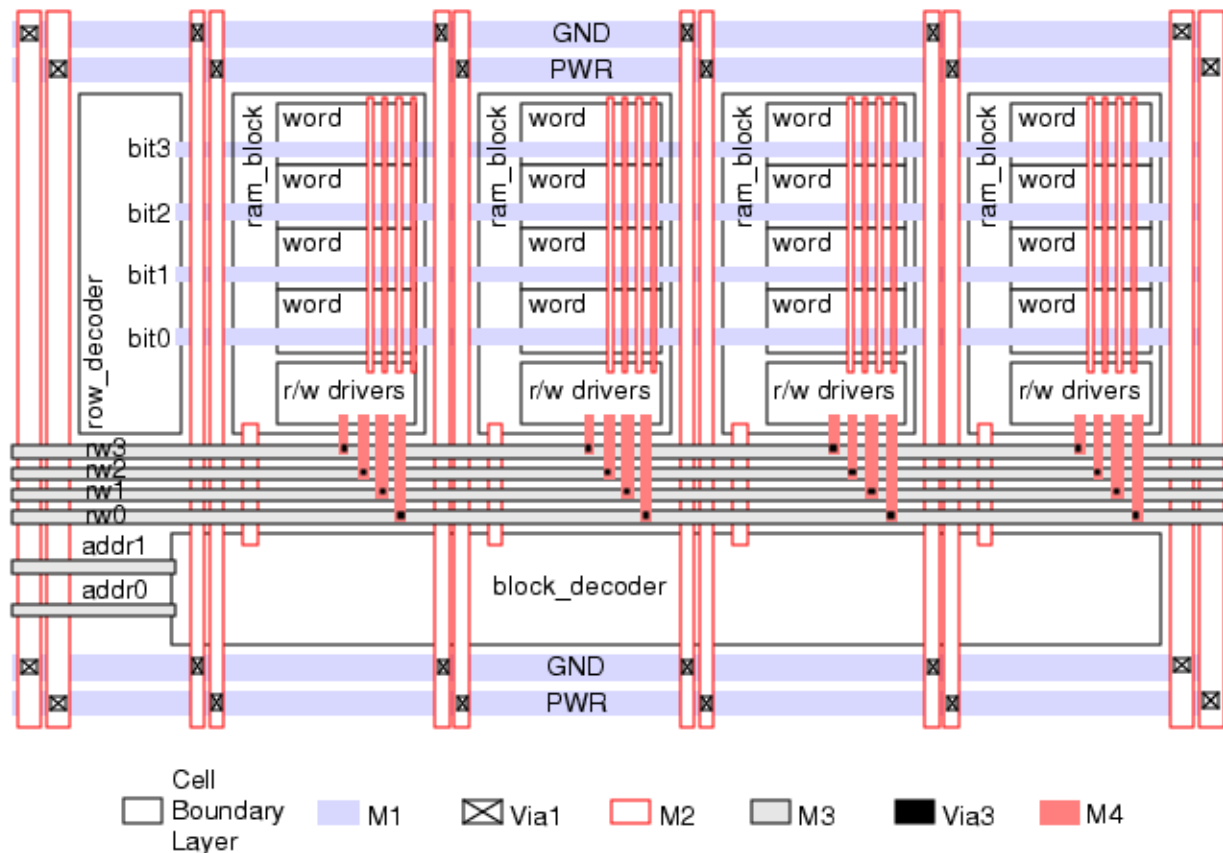
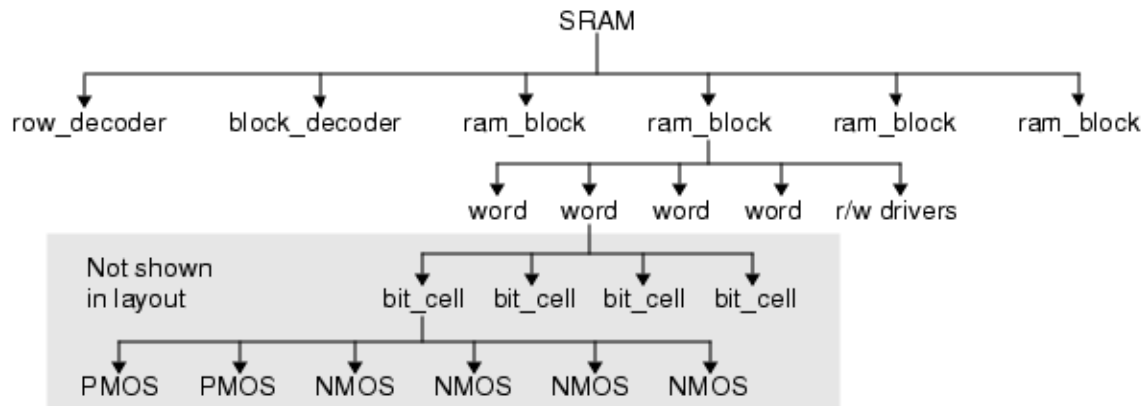


Figure 4-2 shows the design hierarchy for the SRAM block.

Figure 4-2. SRAM Block Design Hierarchy



Hybrid xACT 3D/Rule-Based Extraction

Hybrid xACT 3D/Rule-Based extraction (-3d -hybrid) extracts all nets selected for Calibre xRC and a subset of nets defined for Calibre xACT 3D, and combines the results into a single parasitic database. Use this flow for high-accuracy extraction of critical nets in a full-chip sign-off process.

The command line option -hybrid used with the -3d option for Calibre xACT 3D, activates the hybrid xACT 3D/Rule-Based flow. Nets to be extracted with Calibre xRC are selected with the [PEX Extract Include](#) SVRF statement and the -select command line option. Nets to be excluded from extraction are specified with the [PEX Extract Exclude](#) SVRF statement. The subset of nets to be extracted by Calibre xACT 3D are defined with [PEX Fieldsolver Mode NETS](#) SVRF statement. The extraction results from both tools are combined into a single PDB. The -hybrid option is not supported with the -full option.

Hierarchical Memory Extraction

Hierarchical memory extraction (-xcell -full) extracts data for each user-defined cell as well as the top level of the design. You can nest any number of cells making it ideal for memory arrays. Each cell is extracted only once, which shortens extraction time and netlist size.

The output is a netlist containing hierarchical levels matching the xcell levels. The tool defines each xcell in the final output netlist only once and models each net in the xcell. The resulting netlist is suitable for use with a hierarchical simulator.

In contrast to gate-level extraction, where the output includes netlist data only to the level of the cell's boundary, hierarchical memory extraction includes netlist data within the xcells. The

n-level netlist contains instantiated xcells with subcircuit definitions. Additionally, this type of extraction:

- flattens nets beginning in intermediate non-xcell hierarchical levels into the closest parent xcells
- extracts the successive hierarchical xcell levels until it reaches the primitive device level

Note



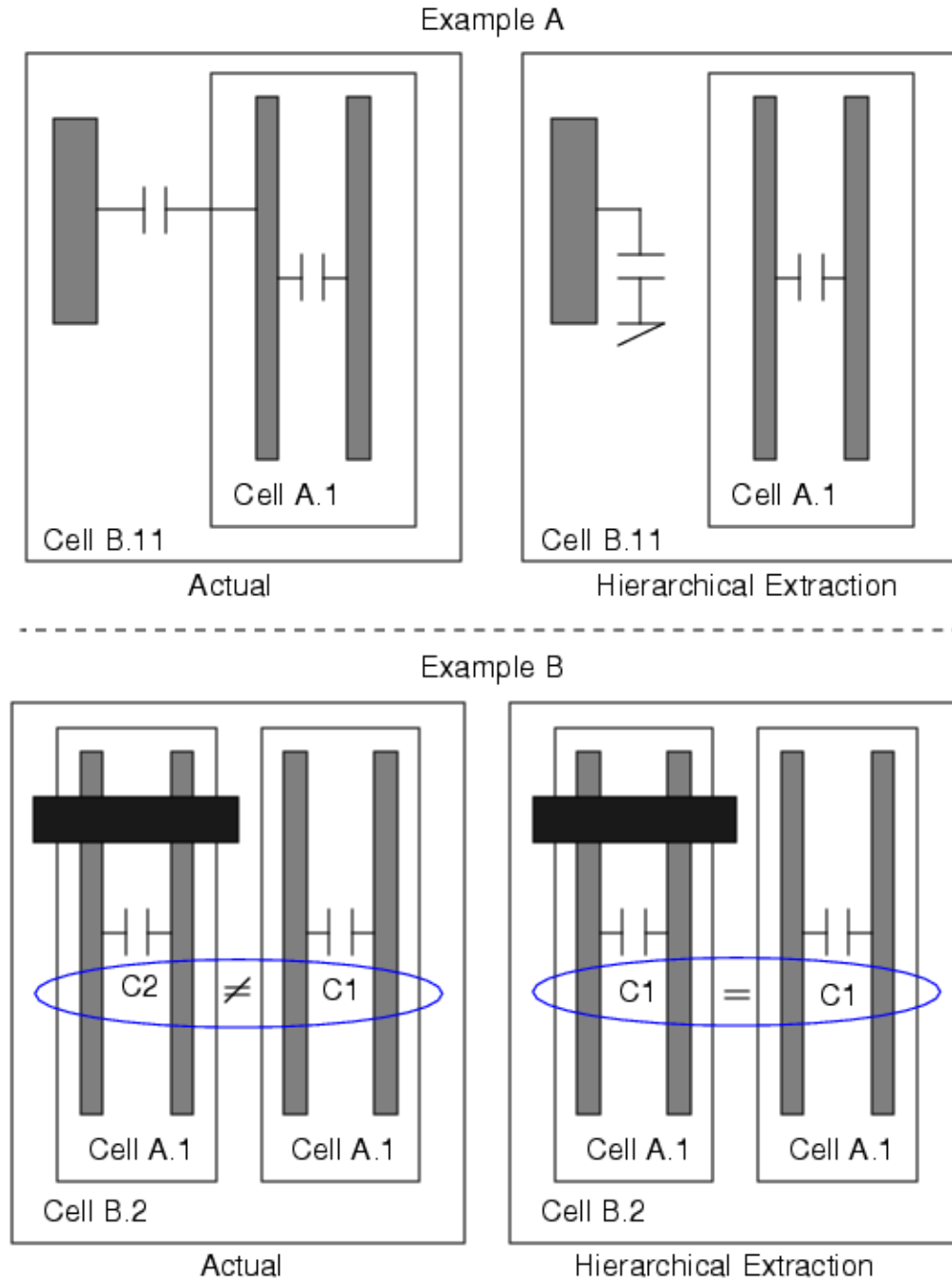
Hierarchical memory extraction optimizes extraction runtimes and netlist size. The actual capacitance values for a net extracted hierarchically and flat will be slightly different.

Because data is stored, analyzed, and processed once per cell instead of once for every flat placement of the cell, hierarchical RC or RCC netlisting cannot show the actual effect of geometries that overlap or abut each specific placement of the cell. (Using the -C flag, you can show the effects for a single specific placement, which is then used for all instances.)

Figure 4-3 shows two examples of how information in a hierarchical netlist approximates the actual layout.

Example A shows the actual RC interaction between a cell, Cell A.1, and an adjacent geometry. It also shows how the same construction looks in a hierarchical netlist. Because Cell A.1 is an xcell and thus context-free by default, the RC component is shown between the adjacent geometry and ground.

Figure 4-3. Hierarchical vs. Actual RC Network Examples



Example B shows how the actual RC component of two identical cells, Cell A.1, compares when a metal layer is placed over only one of the cells. It also shows how the same construction looks in a hierarchical netlist. The actual RC component of the two cells differs, but because Cell A.1 is predefined, the RC component of the two cells is equal in the hierarchical netlist, despite the overlaying metal layer.

Related Topics

[Running Full Hierarchical and Mixed-Signal Hierarchical Extraction](#)

Mixed-Signal Hierarchical Extraction

Mixed-signal hierarchical extraction generates a parasitic netlist for mixed-signal designs.

You may designate two distinct types of xcells in the xcell list:

- Cells that are extracted hierarchically at the transistor level. These cells are extracted and netlisted as subcircuits in the parasitic netlist.
- Cells that are treated as primitive cells. Primitive cells are identified by -P flags in the xcell list. For these cells, no parasitics are extracted and the contents are not netlisted. They are instantiated in the netlist as cell references.

In both cases the netlist preserves a level of hierarchy for the listed cell. For more information on the xcell file, see [“Hierarchy Control with Xcells”](#).

Related Topics

[Running Full Hierarchical and Mixed-Signal Hierarchical Extraction](#)

In-Context Extraction

In-context extraction (-xcell -incontext or -xcell -full) extracts a cell’s parasitic information with reference to structures outside the cell boundary. The location or “context” of a cell will affect the internal parasitics of that cell. In other words, the metal interconnect in the region outside of the cell will affect the capacitances internal to the cell.

In-context extraction allows you to pick a representative cell location and extract its particular parasitic values including the effects of the surrounding material. This is particularly useful for simulating a regular structure such as a memory cell by itself while reflecting the effects of a common placement. This is done by specifying a particular instance of the cell in the xcell file by appending “-C *inst_id*” after the cell name.

The xcell file may contain a mix of regular (context-free) and instance-specific (in-context) cells. (It may also contain primitive cells.)

The contents of the produced PDB depend on how the extraction was invoked, as shown in Table 4-1.

Table 4-1. In-Context PDB Contents

Invocation Switches	PDB Contents
-xcell <i>xcell_file</i>	Regular gate-level or mixed-signal hierarchical extraction. Context IDs are ignored.

Table 4-1. In-Context PDB Contents (cont.)

Invocation Switches	PDB Contents
-xcell <i>xcell_file</i> -incontext	Only the cells specified with context IDs are extracted and any cells they reference that are identified in the xcell list. <i>The primary cell is not extracted.</i> Use the -incontext flag when you want to simulate a specific cell or to replace a context-free cell with a context-aware one.
-xcell <i>xcell_file</i> -full	Similar to full hierarchical extraction, except that cells specified with context IDs have parasitic models reflecting the effects of the surrounding layout. Unlike “-xcell -incontext”, the primary cell is extracted as normal.

With in-context extraction, multiple instances of a cell cannot be extracted into the same PDB. If multiple entries in the xcell file match a cell name, the first entry is used.

Any cells not marked with specific instances are analyzed separately from the environment as with the other types of extraction.

Related Topics

[Discovering Layout Paths for In-Context Cells](#)

Gate-Level Extraction

Gate-level extraction (-xcell) extracts global nets down to top-level cells (logic gates or blocks), which you specify using an xcell list.

When you identify a cell using an xcell list, the Calibre xRC tool extracts the parasitics down to the top-level cell, preserving the cell's internal structure and hierarchy. The output is a netlist with the following two hierarchical levels:

- Top-level cell
- Instances for the highest-level xcells

You cannot use flat LVS for the PHDB with gate-level extraction.

The highest hierarchical xcell instances define the hierarchy. In gate-level extraction xcell instances do not contain other xcell instances. The Calibre xRC tool ignores the connections within the xcell instances and their lower-level structures (cells, transistors, and nested xcells).

Unlike flat extraction, gate-level netlists include net data to the level of the cell's boundary. Nets that cross the cell's boundary are flattened into the parent. You generally use gate-level extraction when you have externally defined libraries containing standard cell data. In this case, include your standard cells in your xcell list.

Related Topics

[Running Gate-Level Extraction](#)

Flat Transistor-Level Extraction

Transistor-level extraction flattens the design's interconnect nets into a top-level cell. These designs are typically less than 500,000 transistors and, subsequently, produce either lumped or distributed net models suitable for resistance and capacitance net models. The Calibre xRC tool creates net models for your design in the form of netlists and models parasitic capacitance and resistance you use for input to analysis tools.

When using this type of extraction, the Calibre xRC tool selects the nets in a design and flattens them to the top-level cell. The Calibre xRC tool extracts the nets in their entirety — from device pin to device pin. Typically, the devices are transistors.

Netlists created from a PDB generated with flat extraction do not have any subcircuits.

Flat transistor-level extraction is the default. If the invocation does not contain “-xcell” when creating the PDB from the command line, a flat PDB is created and the extraction includes the effects of all geometries.

Chapter 5

Producing Parasitic Models

The Calibre xRC tool produces the parasitic model types: Lumped Capacitance, Distributed Resistance, Distributed Resistance and Capacitance, Distributed Resistance and Coupled Capacitance.

For a more general discussion of parasitic effects, see “[Parasitic Effects and Calibre Tools](#).”

Parasitic Devices	49
Types of Parasitic Models	50
Lumped Capacitance.	50
Distributed Resistance	52
Distributed Resistance and Capacitance	52
Distributed Resistance and Coupled Capacitance	54

Parasitic Devices

There are several sources of parasitic effects in a design.

Parasitic effects include:

- **Intrinsic capacitance** — Capacitance between a wire and the substrate (ground).
- **Coupled capacitance** — Capacitance between two conductors (typically interconnects).
- **Parasitic resistance** — Resistance found as an inherent property of any conductive material.

For details about device models, see “[Parasitic Effects and Calibre Tools](#)”.

Types of Parasitic Models

Parasitic models are generally divided into two main types: lumped and distributed.

A lumped parasitic model shows all the parasitic effects as a single element. Distributed parasitic models break up (distribute) the effects more evenly. Distributed parasitic effects can represent either resistance only (referred to as R-only), resistance and capacitance (referred to as RC) or resistance and coupled capacitance (referred to as RCC). When you use RC, coupled capacitance is included with the parasitic capacitance to ground.

The Calibre xRC tool produces the following parasitic model types:

- [Lumped Capacitance](#)
- [Distributed Resistance](#)
- [Distributed Resistance and Capacitance](#)
- [Distributed Resistance and Coupled Capacitance](#)

With the separately licensed Calibre xL extension, the Calibre xRC software can also model inductance. For more information on extracting parasitic inductance, see the [Calibre xL User's Manual](#).

Lumped Capacitance

With lumped capacitance extraction, all capacitance for a net is modeled as one parasitic capacitor between the net and substrate. Depending on the command line options, the parasitic capacitor may include the effects of coupled capacitance to another net with the effect lumped to ground. Resistance is *not* modeled.

[Figure 5-1](#) shows a simplified layout of two cells with interconnect. For lumped capacitance, or Lumped C, two possible models can be extracted based on a command line option in the formatting step.

Figure 5-1. Simplified Layout for Lumped Capacitance

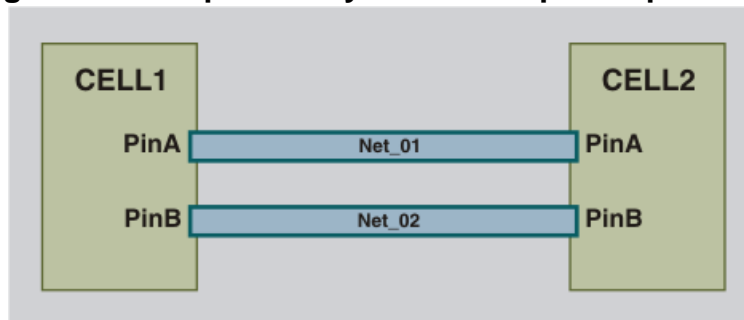


Figure 5-2 shows how the capacitors are represented in the model when the -g command line option is used in the formatting step:

```
calibre -xrc -pdb -c
calibre -xrc -fmt -c -g
```

With this option, the coupled capacitance is added to each intrinsic capacitor, extracted from the net to ground.

Figure 5-2. Lumped Capacitance With Coupled Capacitor Added to Intrinsic Capacitors

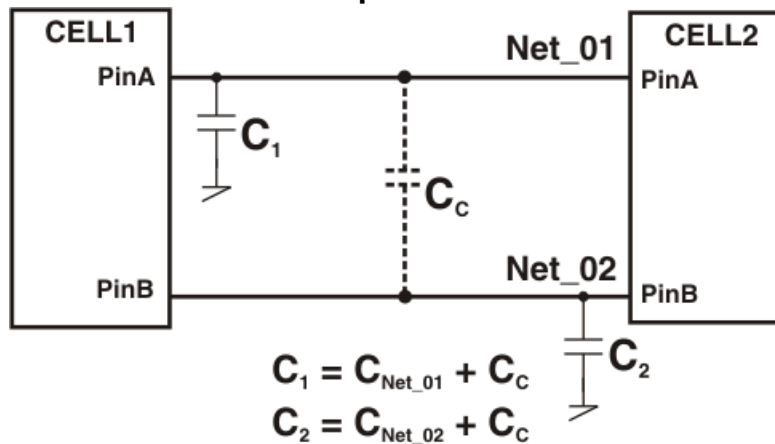
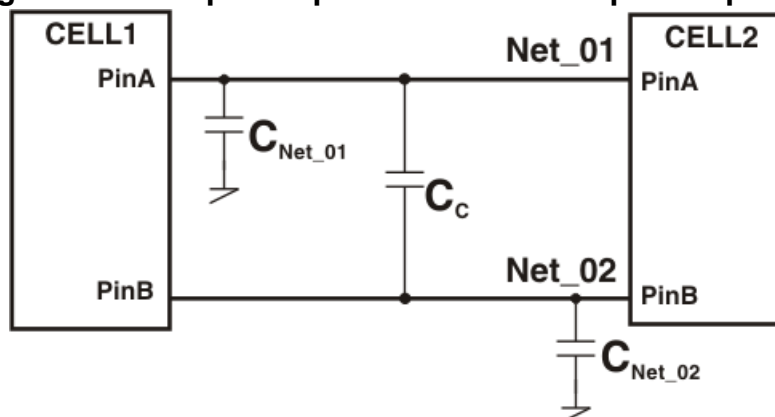


Figure 5-3 shows how the extracted capacitors are represented in the model without the -g command line option:

```
calibre -xrc -pdb -c
calibre -xrc -fmt -c
```

Without the -g option, the couple capacitor is present as a separate device in the net model.

Figure 5-3. Lumped Capacitance With Coupled Capacitor



Distributed Resistance

With distributed resistance extraction, the parasitic resistance of the net is broken into segments representing geometric regions. Capacitance is *not* modeled.

Figure 5-4 shows a simplified layout example with the equivalent distributed resistance extraction model. For Net_01, the net sections A to C represent how Calibre xRC has segments a net for distributed R extraction. This is also the case for Net_02.

Figure 5-4. Simplified Layout for Distributed Resistance

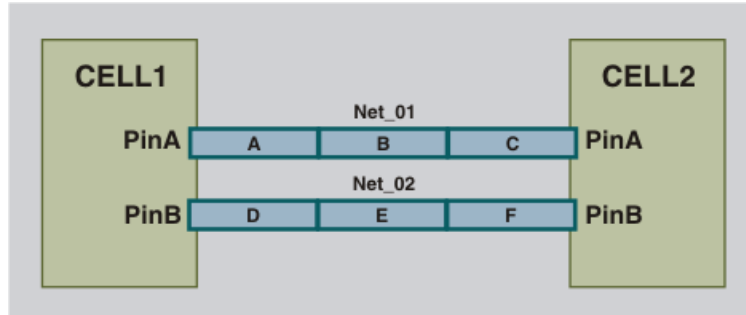
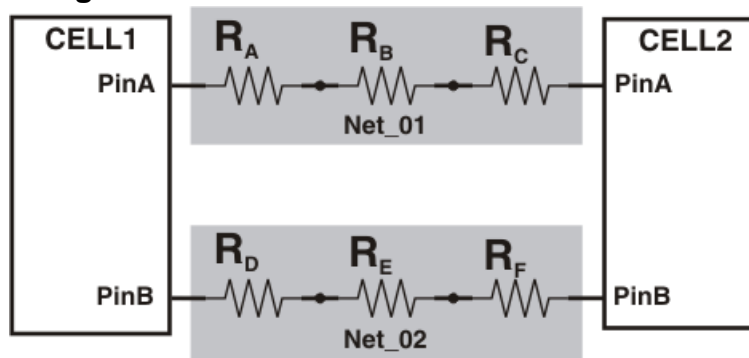


Figure 5-5 shows the extracted net model for distributed R, using the following command lines:

```
calibre -xrc -pdb -r  
calibre -xrc -fmt -r
```

Figure 5-5. Distributed Resistance Extraction



Distributed Resistance and Capacitance

With distributed resistance and capacitance extraction, the parasitic resistance of the net is broken into segments representing geometric regions. The parasitic capacitance is likewise divided into “local” segments going to the substrate, including the effect of coupled capacitance.

Figure 5-6 shows a simplified layout example for the distributed resistance and capacitance parasitic model. If you do not exclude devices for which you supply the models, your final netlist may double-count the parasitic capacitance.

Figure 5-6. Simplified Layout for Distributed Resistance and Capacitance

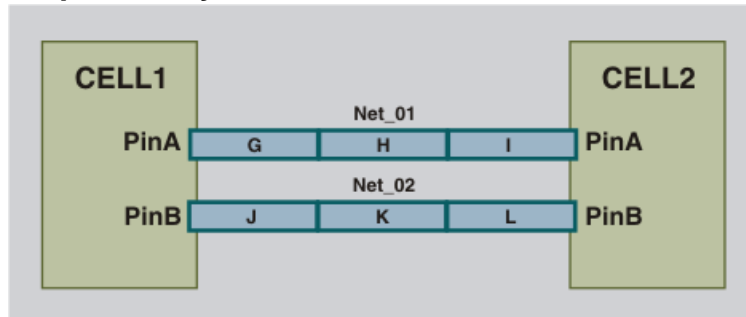


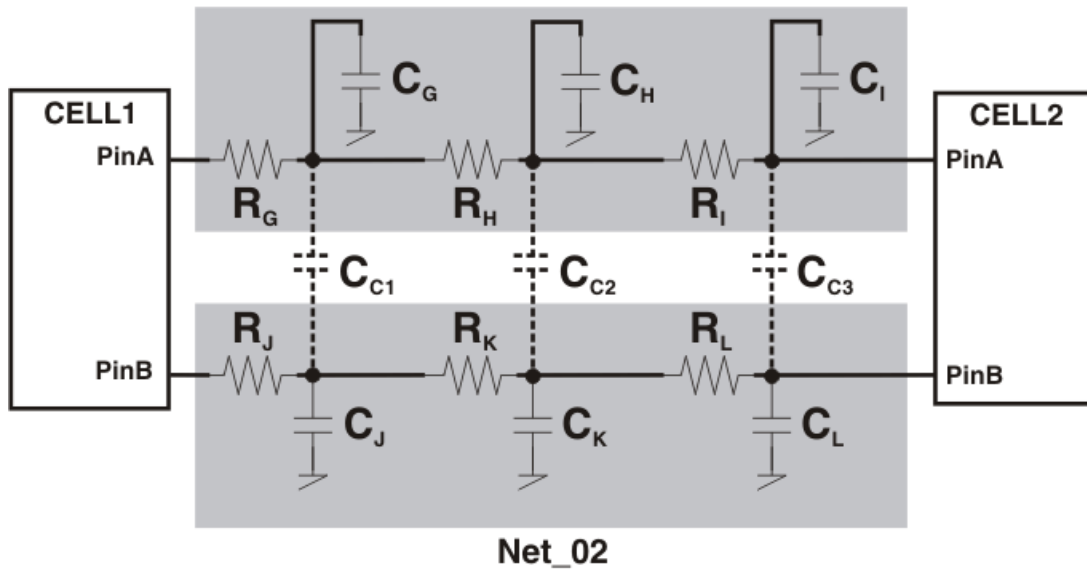
Figure 5-7 shows the extracted net model for distributed R and C, using the following command lines:

```
calibre -xrc -pdb -rc
calibre -xrc -fmt -g [ -rc | -all ]
```

In the formatting step, the option -all is used for writing distributed results; it does not produce lumped capacitance.

Figure 5-7 also shows how the intrinsic capacitors for each net segment include the effect of the coupled capacitor, for example, capacitor CG is the sum of the intrinsic capacitance for segment G plus the coupled capacitance between segment G of Net_01 and segment J of Net_02. Each net is also shaded separately to show that they are not explicitly coupled together.

Figure 5-7. Distributed Resistance and Capacitance Extraction
Net_01



$$\begin{aligned} C_G &= C_{\text{Segment}_G} + C_{C1} & C_H &= C_{\text{Segment}_H} + C_{C2} & C_I &= C_{\text{Segment}_I} + C_{C3} \\ C_J &= C_{\text{Segment}_J} + C_{C1} & C_K &= C_{\text{Segment}_K} + C_{C2} & C_L &= C_{\text{Segment}_L} + C_{C3} \end{aligned}$$

Distributed Resistance and Coupled Capacitance

With distributed resistance and coupled capacitance, the parasitic resistance of the net is divided into segments; however, parasitic capacitance is modeled with separate elements for coupled capacitance and intrinsic capacitance (capacitance to substrate).

Figure 5-8 shows a simplified layout example for the distributed resistance and coupled capacitance parasitic model.

Figure 5-8. Simplified Layout for Distributed Resistance With Coupled Capacitance

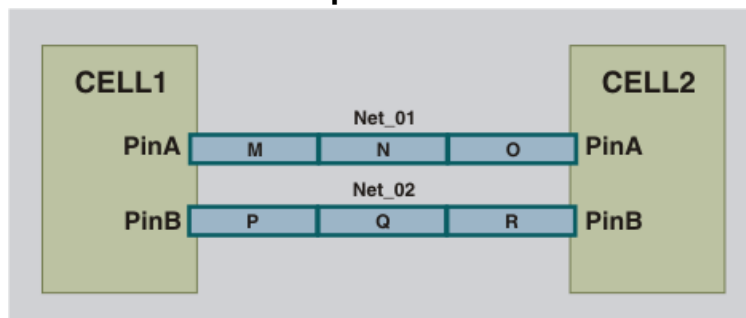
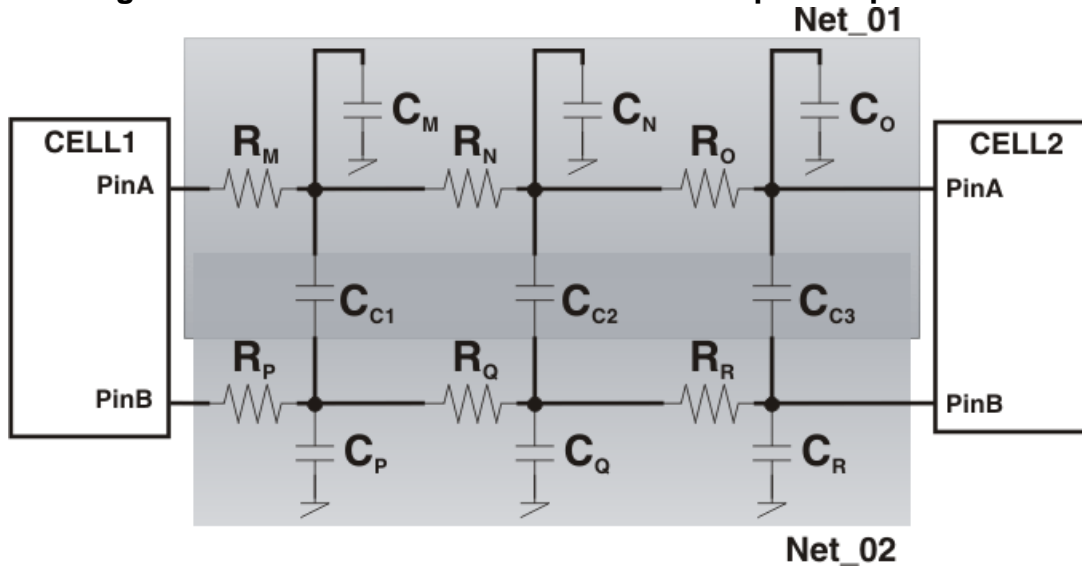


Figure 5-9 shows the extracted net model for distributed R with coupled C when you use the following commands:

```
calibre -xrc -pdb -rcc
calibre -xrc -fmt [ -rcc | -all ]
```

In the formatting step, -all option is used for writing distributed results; it does not produce lumped capacitance.

Figure 5-9. Distributed Resistance With Coupled Capacitance



The overlapping shaded areas show that Net_01 and Net_02 are capacitively coupled through CC1 to CC3. These coupled capacitors are explicitly included in the model and are not added to the intrinsic capacitors, as in the distributed R and C case.

Chapter 6

Basic Extraction Methods

This chapter provides procedures for common extraction runs using both the command-line interface and the Calibre Interactive GUI. Each procedure details the minimum required steps.

Prerequisites for Performing Parasitic Extraction.....	58
Running Transistor-Level Extraction	59
Creating a Transistor-Level Netlist from the Command Line	59
Creating a Transistor-Level Netlist from Calibre Interactive.....	60
Running Gate-Level Extraction	62
Creating a Gate-Level Netlist from the Command Line.....	62
Creating a Gate-Level Netlist from Calibre Interactive	63
Running Full Hierarchical and Mixed-Signal Hierarchical Extraction.....	65
Creating a Hierarchical Netlist from the Command Line.....	65
Creating a Hierarchical Netlist from Calibre Interactive	66
Running ADMS Extraction.....	69
Creating an ADMS Netlist from Calibre Interactive	69
Extracting a Distributed RC PRIMETIME Netlist	76
Creating a PrimeTime Netlist from the Command Line.....	76
Creating a PrimeTime Netlist from Calibre Interactive	77
Extracting a Lumped C Spectre Netlist.....	79
Creating a Capacitance Netlist from the Command Line.....	79
Creating a Spectre Netlist from Calibre Interactive	80
Extracting a Netlist with Mixed Parasitic Networks	82
Mixing Parasitics from the Command Line	82
Mixing Parasitics from Calibre Interactive.....	83
Netlisting a Design Without Parasitics.....	86
Creating an Ideal Netlist from the Command Line.....	86
Creating an Ideal Netlist from Calibre Interactive	87
Backannotating Parasitics to a Source Netlist	89
Backannotating from the Command Line.....	89
Backannotating from Calibre Interactive	90
Generating a Capacitance Summary Report	91
Generating a Net-to-Net Coupling Capacitance Report	92
Reporting Coupled Capacitance from the Command Line.....	92
Reporting Coupled Capacitance from Calibre Interactive	93
Generating a Point-to-Point Resistance Report	95
Reporting Net Resistance from the Command Line.....	95


Reporting Net Resistance from Calibre Interactive	96
Extracting a Placed Cell	98
Extracting Only the Top Level	99
Extracting a Block Using CB	100
Running Calibre xRC CB from the Command Line.	100
Running Calibre xRC CB from Calibre Interactive	101

Prerequisites for Performing Parasitic Extraction

Running Calibre xRC extraction requires that certain conditions be met. These include:

- Geometric database containing the layout.
- The design must be LVS clean, using the same device and connectivity definitions as will be used for parasitic extraction.
- A Calibre xRC or Calibre xRC CB license. See “[Licensing: Parasitic Extraction Products](#)” in the *Calibre Administrator’s Guide* for details.
- SVRF file for the type of parasitics and layout. In addition to your Calibre nmLVS information, the file should contain the following:
 - [Capacitance Order](#) statement.
 - Parasitic calculation statements, usually provided as a separate file by the foundry and included by reference.

Note

 If you are using coplanar layers, the [Capacitance Order](#) statement should be omitted from rule files generated by 2010.3 or later versions of xCalibrate.

For command-line execution, the file must also include the following statements:

- [PEX Netlist](#) to set language and optional content.
- [Layout System](#), [Layout Path](#), and [Layout Primary](#) to identify the top of the design.

Running Transistor-Level Extraction

In Transistor-level extraction devices and nets are examined without regards to cell boundaries. It is the most accurate form of standard xRC extraction, but also the most resource intensive.

Creating a Transistor-Level Netlist from the Command Line	59
Creating a Transistor-Level Netlist from Calibre Interactive.	60

Creating a Transistor-Level Netlist from the Command Line

Transistor-level netlists do not have any specific required SVRF statements or environment variables. They also do not require hcell or xcell files.

Prerequisites

- Layout database that is LVS-clean.
- A valid PEX rule file for this layout.

For more information refer to “[Prerequisites for Performing Parasitic Extraction](#)”.

Procedure

1. Build database of intentional devices.

To use source names, use Calibre nmLVS-H:

```
calibre -lvs -hier -spice $svdb_dir/top_cell.sp rules
```

To use layout names, use Calibre xRC:

```
calibre -xrc -phdb rules
```

2. Extract parasitic effects.

```
calibre -xrc -pdb -parasitic_switch rules
```

where *parasitic_switch* indicates the type of parasitics to extract. For example, -rc for distributed resistance and capacitance or -c for lumped capacitance.

3. Generate the netlist.

If *parasitic_switch* included resistance, use the following:

```
calibre -xrc -fmt -all rules
```

To produce a netlist that only contains lumped capacitance, use the following:

```
calibre -xrc -fmt -c rules
```

Results

Successful Calibre xRC transcripts conclude with a count of errors and warnings as shown.

```
CALIBRE xRC WARNING / ERROR Summary
-----
xRC Warnings   = 2
xRC Errors     = 0
=====
```

If there are no errors, the directory also contains the transistor-level netlist you specified in the PEX Netlist statement. The exact number of files depends on the output format.

Related Topics

[Calibre xRC Tool Invocation Reference](#)

[Netlists](#)

Creating a Transistor-Level Netlist from Calibre Interactive

This procedure specifies the settings particular to running transistor-level extraction from Calibre Interactive.

Prerequisites

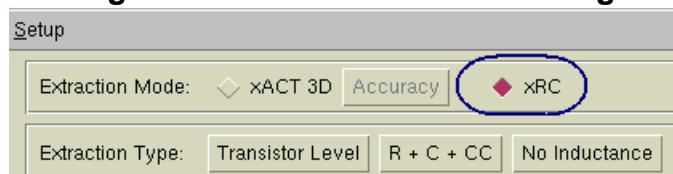
- Layout database that is LVS-clean.
- A valid PEX rule file for this layout.
- For more information refer to “[Prerequisites for Performing Parasitic Extraction](#)”.

Procedure

1. Start the PEX interface in Calibre Interactive.
2. Load a runset or rulefile.
3. Specify the extraction mode.

Click the **Outputs** button in the left pane. Set Extraction Mode to xRC.

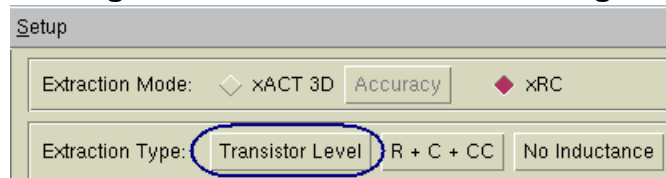
Figure 6-1. Extraction Mode Setting



4. Specify the extraction type.

Click the **Outputs** button in the left pane. In the area above the tabs, set Extraction Type to **Transistor Level**.

Figure 6-2. Transistor Level Setting



5. Set other controls as needed. When ready, click the **Run PEX** button in the left pane.

You do not need to clear the H-Cells field. Because extraction type is set to transistor level, the H-Cells settings are ignored.

Results

Check the Transcripts pane to verify the run completed with no errors. If you have selected “View netlist after PEX finishes” in the Outputs pane, a text viewer appears with the generated netlist loaded.

Related Topics

[Getting Started: Parasitic Extraction Using Calibre Interactive](#)

Running Gate-Level Extraction

The “gates” in gate-level extraction refer to logic gates. These and other predefined circuits are listed in the xcell file. Cells listed in the xcell file appear in the netlist as subcircuit instances.

Note



If an xcell file is empty or the cell names do not match the input, the resulting netlist is flat.

Creating a Gate-Level Netlist from the Command Line **62**

Creating a Gate-Level Netlist from Calibre Interactive **63**

Creating a Gate-Level Netlist from the Command Line

This procedure specifies the settings particular to running transistor-level extraction from the command line.

Prerequisites

- Hcell file or [Hcell](#) statement that includes all cells also listed in the xcell file.
- Xcell file listing cells which will not undergo parasitic extraction. All entries are treated as primitives. You can re-use the hcell file.
- Layout database that is LVS-clean.
- A valid PEX rule file for this layout.

For more information refer to “[Prerequisites for Performing Parasitic Extraction](#)”.

Procedure

1. Flags that are specific to gate-level extraction are in bold.
2. Build database of intentional devices.

To use source names, use Calibre nmLVS-H:

```
calibre -lvs -hier -hcell hcell_file -spice $svdb_dir/top_cell.sp  
rules
```

To use layout names, use Calibre xRC:

```
calibre -xrc -phdb -hcell hcell_file rules
```

3. Extract parasitic effects.

```
calibre -xrc -pdb -xcell xcell_file -parasitic_switch rules
```

where *parasitic_switch* indicates the type of parasitics to extract. For example, -rc for distributed resistance and capacitance.

4. Generate the netlist. You do not need to specify the xcell list.

```
calibre -xrc -fmt rules
```

Results

Successful Calibre xRC transcripts conclude with a count of errors and warnings as shown.

```

CALIBRE xRC WARNING / ERROR Summary
-----
xRC Warnings   = 2
xRC Errors     = 0
=====

```

If there are no errors, the directory also contains the transistor-level netlist you specified in the PEX Netlist statement. The exact number of files depends on the output format.

Related Topics

[Calibre xRC Tool Invocation Reference](#)

[Netlists](#)

[Gate-Level Extraction](#)

Creating a Gate-Level Netlist from Calibre Interactive

This procedure specifies the settings particular to running gate-level extraction from Calibre Interactive.

Prerequisites

- Hcell file or [Hcell](#) statement that includes all cells also listed in the xcell file.
- Xcell file listing cells which will not undergo parasitic extraction. All entries are treated as primitives. You can re-use the hcell file.
- Layout database that is LVS-clean.
- A valid PEX rule file for this layout.

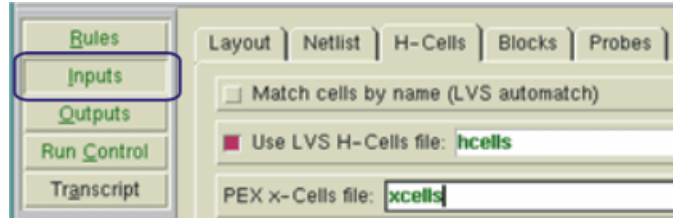
For more information refer to “[Prerequisites for Performing Parasitic Extraction](#)”.

Procedure

1. Start the PEX interface in Calibre Interactive:
2. Load a runset or rulefile.

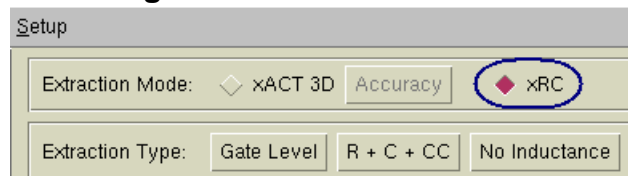
3. Specify the hcells, xcells, and extraction type.
 - a. Click the **Inputs** button in the left pane. Select the **H-Cells** tab. Specify the hcell and xcell files in the appropriate fields. (They can be the same file.)

Figure 6-3. Specifying HCell and XCell Files in Calibre Interactive



- b. Click the **Outputs** button in the left pane. Set Extraction Mode to **xRC**.

Figure 6-4. Extraction Mode



- c. In the area above the tabs, set Extraction Type to **Gate Level**.

Figure 6-5. Gate Level Setting



4. Set other controls as needed. When ready, click the **Run PEX** button in the left pane.

Results

Check the Transcripts pane to verify the run completed with no errors. If you have selected “View netlist after PEX finishes” in the Outputs pane, a text viewer appears with the generated netlist loaded.

Related Topics

[Getting Started: Parasitic Extraction Using Calibre Interactive](#)


[Gate-Level Extraction](#)

[Types of Extraction](#)

Running Full Hierarchical and Mixed-Signal Hierarchical Extraction

The difference between full hierarchical extraction and mixed-signal hierarchical extraction is whether or not all xcells are extracted in full and independent of the layout. The xcell file specifies whether each cell is a “regular” cell (no flag or a -c) to be extracted in full or a primitive cell (-p) and not extracted. The invocation and SVRF file are identical.

Note

 If an xcell file is empty or the cell names do not match the input, the resulting netlist is flat.

Full hierarchical extraction is intended for use on designs with significant amounts of repeated hierarchy such as memory.

Creating a Hierarchical Netlist from the Command Line	65
Creating a Hierarchical Netlist from Calibre Interactive	66

Creating a Hierarchical Netlist from the Command Line

This procedure describes how to create a hierarchical netlist from the command line.

Prerequisites

- A valid PEX rule file for this layout. If you are creating a DSPF netlist for use in a hierarchical simulator that accepts a position file, add [PEX Netlist Position File](#).
- Hcell file or [Hcell](#) statement that includes all cells also listed in the xcell file.
- Xcell file listing cells to preserve in the parasitic netlist.
- Layout database that is LVS-clean.

For more information refer to “[Basic Extraction Methods](#)”.

Procedure

1. If you are netlisting to DSPF, use the [PEX Netlist](#) statement with the DSPF and HSIM keywords. It improves modeling of feedthrough nets, but only affects DSPF output.
2. Build database of intentional devices. Flags that are specific to hierarchical extraction are in bold.

To use source names, use Calibre nmLVS-H:

```
calibre -lvs -hier -hcell hcell_file -spice $svdb_dir/top_cell.sp
rules
```

To use layout names, use Calibre xRC:

```
calibre -xrc -phdb -hcell hcell_file rules
```

3. Extract parasitic effects.

```
calibre -xrc -pdb -xcell xcell_file -full -parasitic_switch rules
```

where *parasitic_switch* indicates the type of parasitics to extract. For example, -rc for distributed resistance and capacitance.

4. Generate the netlist. You do not need to specify the xcell list.

```
calibre -xrc -fmt -full rules
```

Results

Successful Calibre xRC transcripts conclude with a count of errors and warnings as shown.

```

CALIBRE xRC WARNING / ERROR Summary
-----
xRC Warnings   = 2
xRC Errors     = 0
=====
```

If there are no errors, the directory also contains the netlist you specified in the PEX Netlist statement. Each xcell appears in the netlist as a subcircuit. The exact number of files depends on the output format.

Related Topics

[Calibre xRC Tool Invocation Reference](#)

[Netlists](#)

[Hierarchical Memory Extraction](#)

[Mixed-Signal Hierarchical Extraction](#)

Creating a Hierarchical Netlist from Calibre Interactive

This procedure assumes you have an SVRF file set up for your type of parasitic extraction and only specifies the settings particular to producing hierarchical netlists.

Prerequisites

- A valid PEX rule file for this layout. If you are creating a DSPF netlist for use in a hierarchical simulator that accepts a position file, add [PEX Netlist Position File](#).
- Hcell file or [Hcell](#) statement that includes all cells also listed in the xcell file.

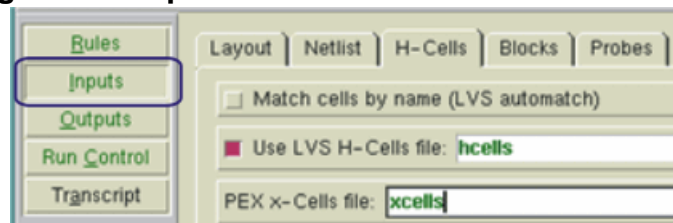
- Xcell file listing cells to preserve in the parasitic netlist.
- Layout database that is LVS-clean.

For more information refer to “[Prerequisites for Performing Parasitic Extraction](#)”.

Procedure

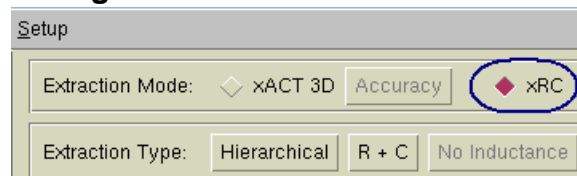
1. Start the PEX interface in Calibre Interactive:
2. Load a runset or rulefile.
3. Specify the hcells, xcells, and extraction type.
 - a. Click the **Inputs** button in the left pane. Select the **H-Cells** tab. Specify the hcell and xcell files in the appropriate fields. (They can be the same file.)

Figure 6-6. Inputs Pane for Hierarchical Extraction



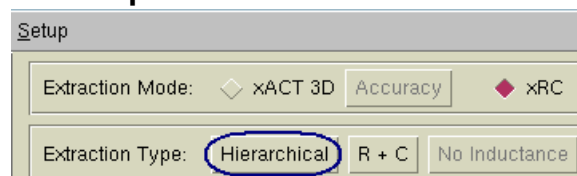
- b. Click the **Outputs** button in the left pane. Set Extraction Mode to **xRC**.

Figure 6-7. Set Extraction Mode



- c. In the area above the tabs, set Extraction Type to **Hierarchical**

Figure 6-8. Outputs Pane for Hierarchical Extraction.



4. Set other controls as needed. When ready, click the **Run PEX** button in the left pane.

Results

Check the Transcripts pane to verify the run completed with no errors. If you have selected “View netlist after PEX finishes” in the Outputs pane, a text viewer appears with the generated netlist loaded.

Related Topics

[Getting Started: Parasitic Extraction Using Calibre Interactive](#)

[Hierarchical Memory Extraction](#)

[Basic Extraction Methods](#)

Running ADMS Extraction

ADMS extraction has been optimized for running with Calibre Interactive to produce netlists that easily work with Siemens EDA ADVance MS simulator.

Creating an ADMS Netlist from Calibre Interactive 69

Creating an ADMS Netlist from Calibre Interactive

This procedure assumes you know how to run the Calibre Interactive GUI.

For a more detailed description of how to use Calibre Interactive, refer to the [Calibre Interactive User's Manual](#).

Prerequisites

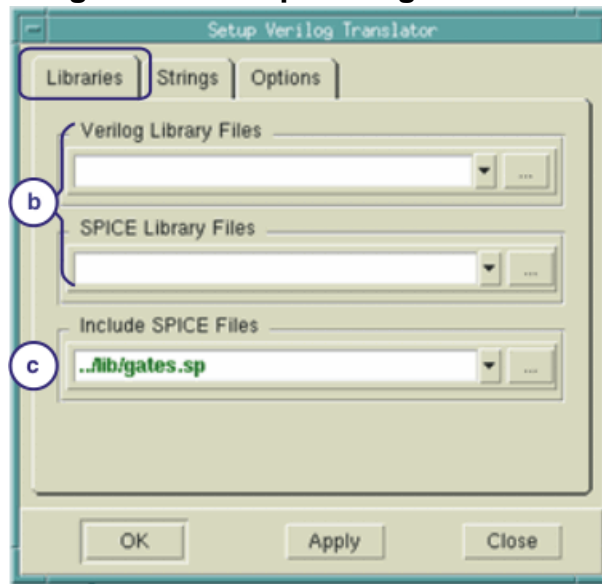
- A Calibre xRC to ADVance MS license in addition to the Calibre xRC or Calibre xRC CB license. See “[Licensing: Parasitic Extraction Products](#)” in the *Calibre Administrator's Guide* for details.
- A valid PEX rule file for this layout.
- Hcell file or [Hcell](#) statement that includes all cells also listed in the xcell file.
- Xcell file listing primitive cells.
- Verilog libraries for the design source.
- A delay calculator, such as Siemens EDA Time-it.
- Layout database that is LVS-clean.

For more information refer to “[Prerequisites for Performing Parasitic Extraction](#)”.

Procedure

1. Start the Calibre Interactive PEX interface:
2. Load a runset or rulefile.
3. Set up the Verilog Translator.
 - a. Select **Setup > Verilog Translator**. The dialog appears as shown in [Figure 6-9](#).
 - b. In the Setup Verilog Translator dialog box, click the **Libraries** tab. Enter the location of the library files in the corresponding Verilog Library Files or SPICE Library Files fields.
 - c. To include SPICE files of lower level subcircuits referenced in the Verilog library file, specify the path in the Include SPICE files field.
 - d. Click **OK** to close the dialog box.

Figure 6-9. Setup Verilog Translator



For information on other options in the Verilog Translator, see “[Setting Up the Verilog Translator \(v2lvs\)](#)” in the *Calibre Interactive User’s Manual*.

4. Set up the delay calculator to produce SDF for the standard cells. The ADMS flow uses Verilog gate level blocks, which are represented in SDF format, to backannotate delays.
 - a. Select **Setup > Delay Calculation**. The dialog appears as shown in [Figure 6-10](#).
 - b. In the Technology Files dialog box, enter the path to your standard cell library.
 - c. In the Time-it MGC_HOME field, enter the path to your Time-it™ MGC_HOME directory. The Time-it tool can be downloaded from Support Center.

Note


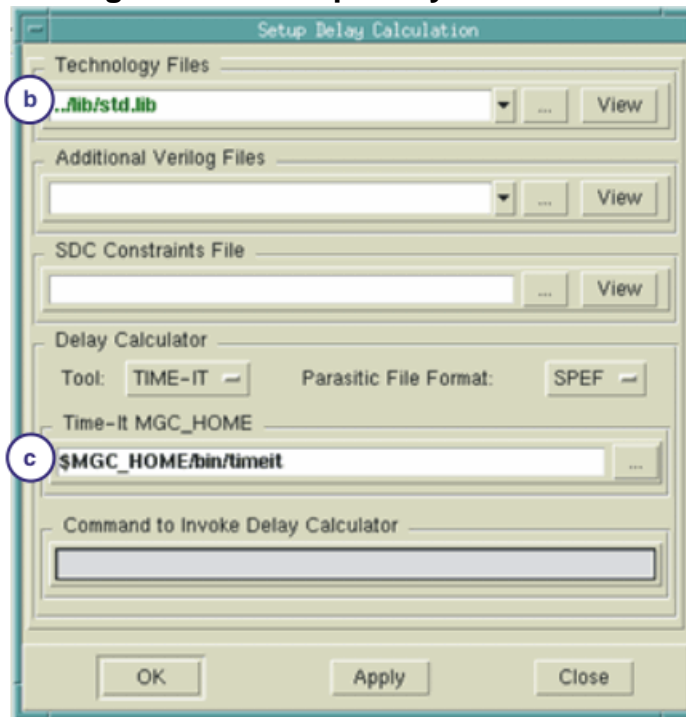
 You can select delay calculators other than Time-it by selecting **Other** from the drop-down menu, and entering the path of your preferred delay calculator in the **Command to Invoke Delay Calculator** field. See “[Setting Delay Calculation](#)” in the *Calibre Interactive User’s Manual* for required parameters.

Figure 6-10. Setup Delay Calculation



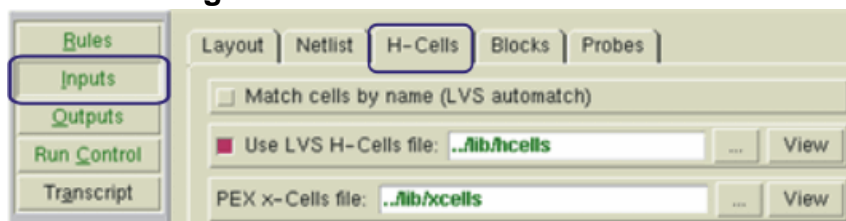
- d. Click **OK** to close the dialog box.
5. In the **PEX Options** pane, define the ground node name (usually VSS) in the **Netlist** tab.
6. In the **Inputs** pane, fill out the **Netlist** tab.
 - a. Click the Format button and select MIXED. This specifies the format of the top level netlist, and exposes both the SPICE and Verilog fields.
 - b. Enter the names of all the source netlist files for LVS and extraction. If a file includes other files, the other files do not need to be separately listed.
 - c. Browse to the name of the top cell in the Top Cell field. (Browsing is recommended, since the cell name is case-sensitive.)

Figure 6-11. Netlist Tab for ADMS



7. Fill out the **H-Cells** tab. The ADMS flow requires an xcell file which identifies primitives.

Figure 6-12. H-Cells Tab for ADMS

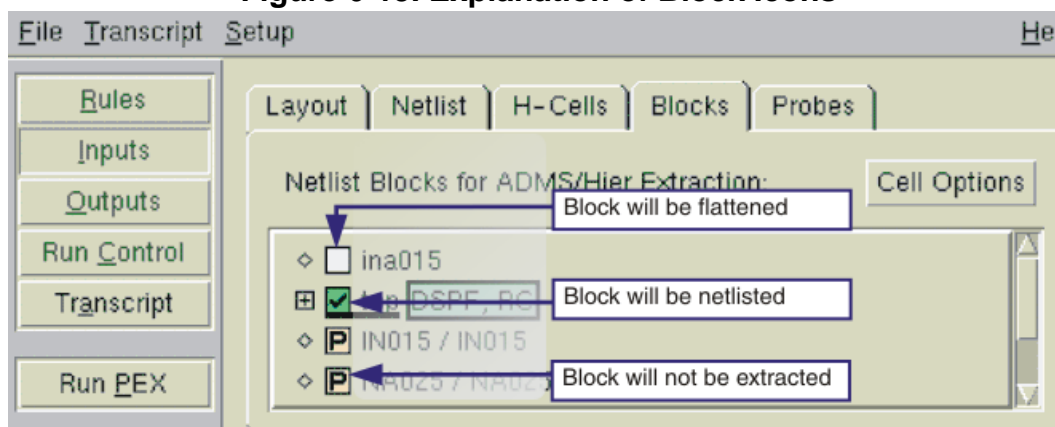


8. Fill out the **Blocks** tab. Here you specify the blocks to be netlisted and the output format to use. The blocks are identified from the source files in the Netlist tab. The xcell file determines whether blocks are primitives.

- a. Click block names to mark them for netlisting or flattening into the parent cell. Blocks marked with a “+” contain other blocks; click the + to show them.

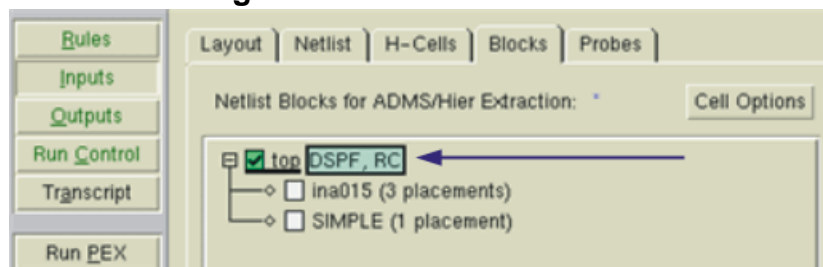
You cannot unselect the top cell; it is always netlisted. Also, you cannot change primitives, marked with a P, from the interface.

Figure 6-13. Explanation of Block Icons



- b. For blocks marked with a green checkmark, right-click on block names and select **Format** to access the netlist formats. (The block must be marked for netlisting first.) Formatted blocks show the format and extraction mode after them, as shown in [Figure 6-14](#).

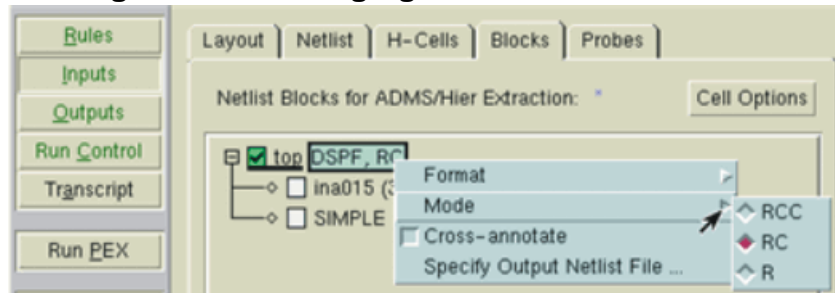
Figure 6-14. Formatted Block



Only formats accepted by the ADVance MS simulator are available. This includes DSPF, Eldo, HSPICE, SPEF, and SDF. SDF is the format for backannotating delays for Verilog gate-level description.

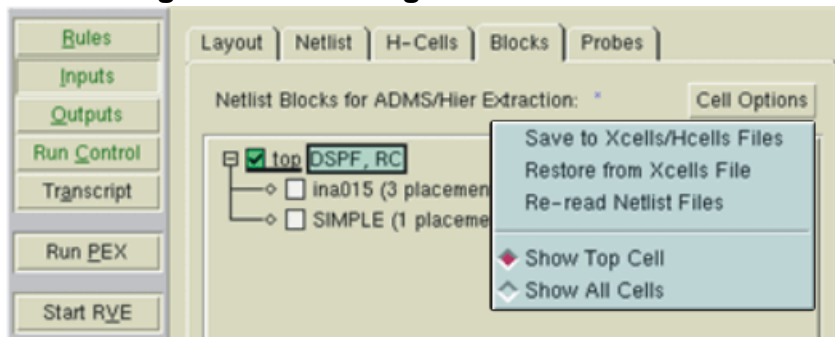
- c. To change from the default RC extraction mode to RCC or R, right-click the green box with the formats. This menu also lets you specify the netlist name and mark a block for cross-annotation.

Figure 6-15. Changing Block Extraction Mode



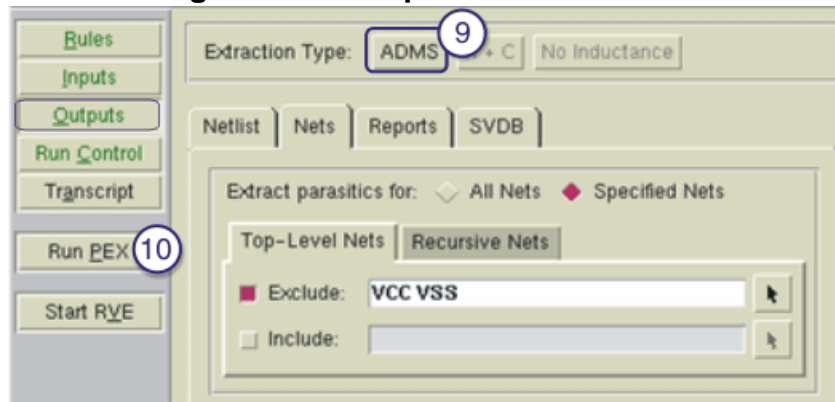
- d. Save your changes by selecting **Cell Options > Save to Xcells/Hcells Files**. Blocks which you have marked for netlisting are added to the hcell and xcell files.

Figure 6-16. Saving Blocks to Cell Files



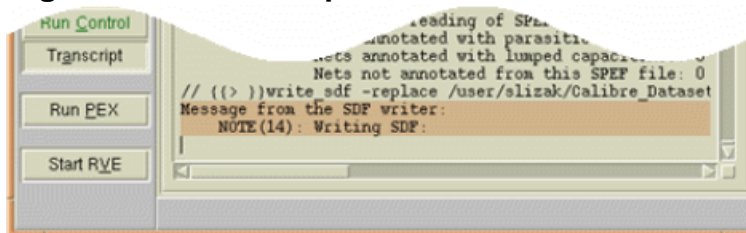
9. In the **Outputs** pane, set Extraction Type to **ADMS**. The other settings are read from the Blocks tab. (Do not change Format here - it will not be reflected in the Blocks tab.) You may also want to exclude power and ground nets from extraction. This is set under the **Nets** tab.

Figure 6-17. Outputs Tab for ADMS



10. Click the **Run PEX** button to extract parasitics and calculate delay. When the transcript shows “Writing SDF”, the run is complete.

Figure 6-18. Transcript for Successful ADMS Run



Results

Running PEX produces several required files, and at least two other files in the run directory. The required files are subsequently included in the ADMS testbench for post-layout simulation of the design.

The required files are:

- *digital_blocks.sdf* — The SDF files of the digital blocks.
- *top_cell.format* — The post-layout netlist of the top level, where *format* is the format specified in the Blocks tab.
- *bind.inc* — A file containing the .BIND statements for ADMS. The .BIND statements will replace the digital block instance within the post-layout netlist with its verilog model.
- *top.inc* — A file specifying the location and name of the top-level parasitic netlist.

The other files are:

- *digital_blocks.spef* — Intermediate files for the digital blocks.
- *digital_blocks.spef.log* — Time-it log files.

Related Topics

[Getting Started: Parasitic Extraction Using Calibre Interactive](#)

[Time-it Tool Overview](#)

Extracting a Distributed RC PRIMETIME Netlist

Distributed RC extraction is useful when the ultimate goal is to analyze circuit loading or timing analysis. This example happens to show transistor-level extraction, but distributed RC and PrimeTime netlist format also work with gate-level and hierarchical extraction.

Creating a PrimeTime Netlist from the Command Line	76
Creating a PrimeTime Netlist from Calibre Interactive	77

Creating a PrimeTime Netlist from the Command Line

Use this procedure to create a PrimeTime netlist from the command line.

The key to creating a PrimeTime format netlist is the PRIMETIME keyword in the PEX Netlist statement. The PRIMETIME keyword is only supported in distributed RC mode, so you must extract parasitics using the -rc parasitic flag.

Prerequisites

- A valid PEX rule file for this layout including a [PEX Netlist](#) statement of the following form:

```
PEX NETLIST netlist_filename DSPF PRIMETIME
```

or

```
PEX NETLIST netlist_filename SPEF PRIMETIME
```

- Layout database that is LVS-clean

For more information refer to “[Basic Extraction Methods](#)”.

Procedure

1. Build database of intentional devices.

To use source names, use Calibre nmLVS-H:

```
calibre -lvs -hier -spice $svdb_dir/top_cell.sp rules
```

To use layout names, use Calibre xRC:

```
calibre -xrc -phdb rules
```

2. Extract parasitic effects.

```
calibre -xrc -pdb -rc rules
```

3. Generate the netlist.

```
calibre -xrc -fmt rules
```

Results

Successful Calibre xRC transcripts conclude with a count of errors and warnings as shown.

```

CALIBRE xRC WARNING / ERROR Summary
-----
xRC Warnings   = 2
xRC Errors     = 0
=====

```

If there are no errors, the directory also contains the DSPF or SPEF netlist you specified in the PEX Netlist statement.

Creating a PrimeTime Netlist from Calibre Interactive

This procedure specifies the settings particular to creating a PrimeTime netlist from Calibre Interactive.

Prerequisites

- A valid PEX rule file for this layout.
- Layout database that is LVS-clean.

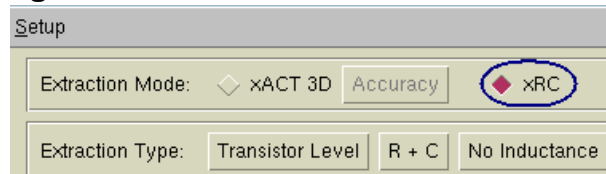
For more information refer to “[Prerequisites for Performing Parasitic Extraction](#)”.

Procedure

1. Start the PEX interface in Calibre Interactive.
2. Load a runset or rulefile.
3. Specify the extraction mode.

Click the **Outputs** button in the left pane. Set Extraction Mode to **xRC**.

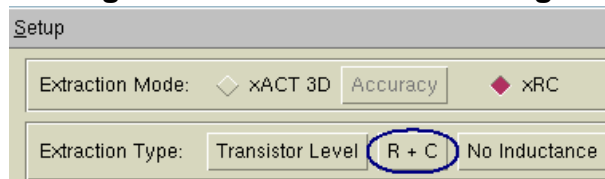
Figure 6-19. xRC Extraction Mode Setting



4. Specify the extraction type.

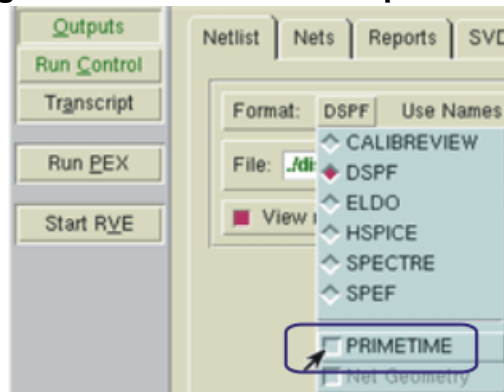
In the area above the tabs, set Extraction Type to **R + C**.

Figure 6-20. R + C Mode Setting



5. Select the **Netlist** tab, and set the format to **DSPF** or **SPEF**. The PRIMETIME item is only available for those netlist formats.
6. Select the format menu again and set the **PRIMETIME** option.

Figure 6-21. PrimeTime Output Setting



7. Set other controls as needed. When ready, click the **Run PEX** button in the left pane.

Results

Check the Transcripts pane to verify the run completed with no errors. If you have selected “View netlist after PEX finishes” in the Outputs pane, a text viewer appears with the generated netlist loaded.

Related Topics

[Getting Started: Parasitic Extraction Using Calibre Interactive](#)

Extracting a Lumped C Spectre Netlist

Lumped capacitance is often the first step in characterizing a circuit. It is also easier to work with for backannotating schematics. This example happens to show transistor-level extraction, but lumped capacitance is available in all the extraction types.

Creating a Capacitance Netlist from the Command Line	79
Creating a Spectre Netlist from Calibre Interactive	80

Creating a Capacitance Netlist from the Command Line

Use this procedure to create a capacitance netlist from the command line.

Prerequisites


- A valid PEX rule file for this layout including a [PEX Netlist](#) statement of the following form:

```
PEX NETLIST netlist_filename SPECTRE
```

- Layout database that is LVS-clean.

For more information refer to “[Basic Extraction Methods](#)”.

Note

 Unlike distributed extraction, lumped C requires -c in the netlist generation step. Using -all or not specifying -c errors out with “ERROR: Resistance requested in netlist, but none was extracted.”

Procedure

1. Build database of intentional devices.

To use source names, use Calibre nmLVS-H:

```
calibre -lvs -hier -spice $svdb_dir/top_cell.sp rules
```

To use layout names, use Calibre xRC:

```
calibre -xrc -phdb rules
```

2. Extract parasitic effects.

```
calibre -xrc -pdb -c rules
```

3. Generate the netlist.

```
calibre -xrc -fmt -c rules
```

Results

Successful Calibre xRC transcripts conclude with a count of errors and warnings as shown.

```
CALIBRE xRC WARNING / ERROR Summary
-----
xRC Warnings   = 2
xRC Errors     = 0
=====
```

If there are no errors, the directory also contains the Spectre format netlist you specified in the PEX Netlist statement. The coupled capacitance is in the .pxi file.

Related Topics

[Calibre xRC Tool Invocation Reference](#)

[Netlists](#)

[Output Reference](#)

Creating a Spectre Netlist from Calibre Interactive

This procedure only specifies the settings particular to creating a lumped-C Spectre netlist.

Prerequisites

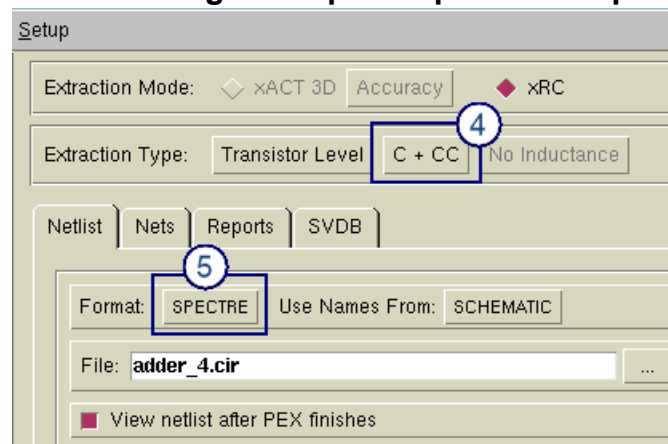
- A valid PEX rule file for this layout.
- Layout database that is LVS-clean.

For more information refer to “[Basic Extraction Methods](#)”.

Procedure

1. Start the PEX interface in Calibre Interactive.
2. Load a runset or rulefile.
3. Specify the extraction mode. Set Extraction Mode to **xRC**.
4. Specify the extraction type. Click the **Outputs** button in the left pane. In the area above the tabs, set Extraction Type to **C + CC**.
5. Select the **Netlist** tab, and set the format to **SPECTRE**.

Figure 6-22. Extracting a Lumped Capacitance Spectre Netlist



6. Set other controls as needed. When ready, click the **Run PEX** button in the left pane.

Results

Check the Transcripts pane to verify the run completed with no errors. If you have selected “View netlist after PEX finishes” in the Outputs pane, a text viewer appears with the generated netlist loaded.

Related Topics

[Getting Started: Parasitic Extraction Using Calibre Interactive](#)

Extracting a Netlist with Mixed Parasitic Networks

Simulations run faster when netlists are simpler and have fewer elements. You can improve your simulation time by extracting only the parasitics you need for the analysis: for example, extracting capacitance throughout but resistance only for time-critical nets. To do this, run the PDB stage multiple times using `-select` as shown below. This is known as *iterative extraction*.

Mixing Parasitics from the Command Line	82
Mixing Parasitics from Calibre Interactive.....	83

Mixing Parasitics from the Command Line

This procedure shows transistor-level extraction, but you could also use it with gate-level or full hierarchical extraction. The extraction level must stay the same throughout.

Prerequisites

- A valid PEX rule file for this layout including a [PEX Extract Include](#) statement in the SVRF file.
- Layout database that is LVS-clean.

For more information refer to “[Basic Extraction Methods](#)”.

Procedure

1. Build database of intentional devices.

To use source names, use Calibre nmLVS-H:

```
calibre -lvs -hier -spice $svdb_dir/top_cell.sp rules
```

To use layout names, use Calibre xRC:

```
calibre -xrc -phdb rules
```

2. Extract the “background” parasitic effects for the whole design.

```
calibre -xrc -pdb -c rules
```

3. Extract the more detailed parasitics for some nets.


```
calibre -xrc -pdb -rccl -select rules
```

Calibre extracts parasitics for only the nets listed in the PEX Extract Include statement because of the `-select` switch. You can edit the statement and run this step as many times as needed. The rest of the SVRF rule file and the layout must remain the same.

4. Generate the netlist.

```
calibre -xrc -fmt rules
```

Caution

 Do not extract more parasitics to the same SVDB directory after generating a netlist. Netlist generation can eliminate internal nodes so adding in new parasitic elements could introduce subtle errors.

Results

Successful Calibre xRC transcripts conclude with a count of errors and warnings as shown.

```

CALIBRE xRC WARNING / ERROR Summary
-----
xRC Warnings    = 2
xRC Errors      = 0
=====

```

If there are no errors, the directory also contains the netlist you specified in the PEX Netlist statement. The exact number of files depends on the output format.

Related Topics

[Calibre xRC Tool Invocation Reference](#)

[Extracting Particular Nets](#)

Mixing Parasitics from Calibre Interactive

To mix parasitics in Calibre Interactive, you need to use some of the advanced options.

Prerequisites

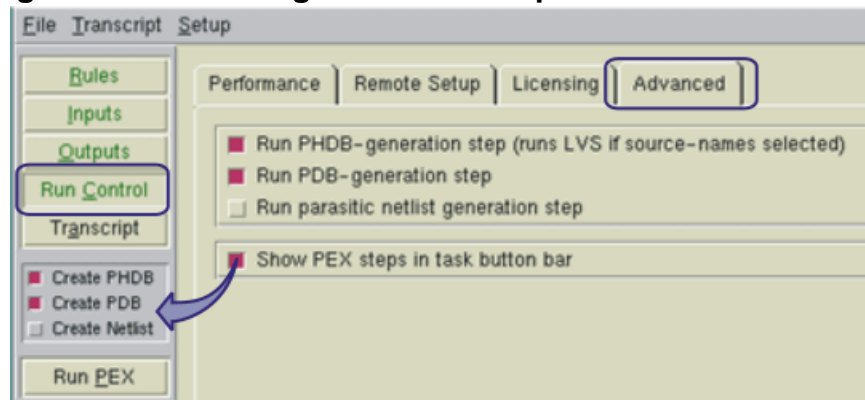
- A valid PEX rule file for this layout.
- Layout database that is LVS-clean.

For more information refer to “[Basic Extraction Methods](#)”.

Procedure

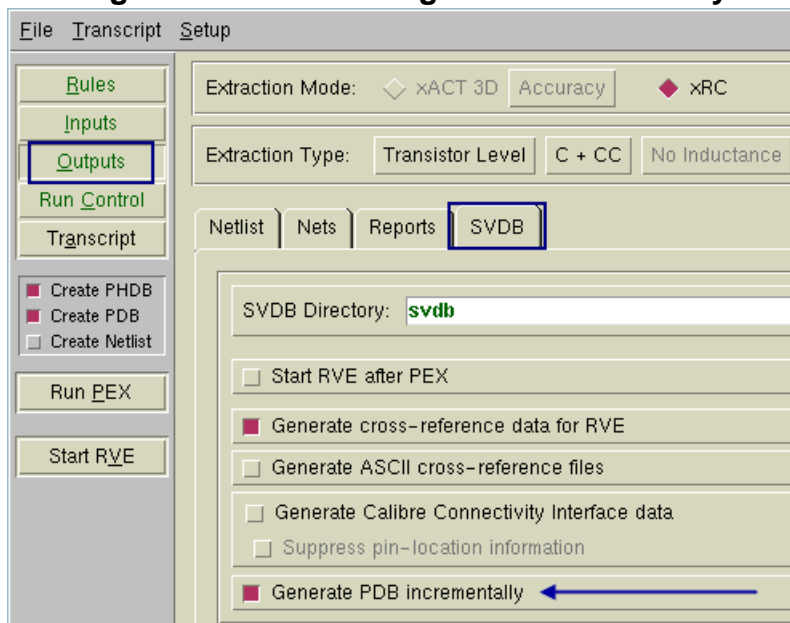
1. Start the PEX interface in Calibre Interactive.
2. Load a runset or rulefile.
3. Enable the PEX step controls.

Figure 6-23. Enabling Extraction Steps in Calibre Interactive



- a. Click the **Run Control** button in the left pane.
- b. Click the **Advanced** tab.
- c. Select **Show PEX steps in task button bar**.
4. Uncheck the Create Netlist step in the PEX step controls.
5. Enable cumulative extraction steps.

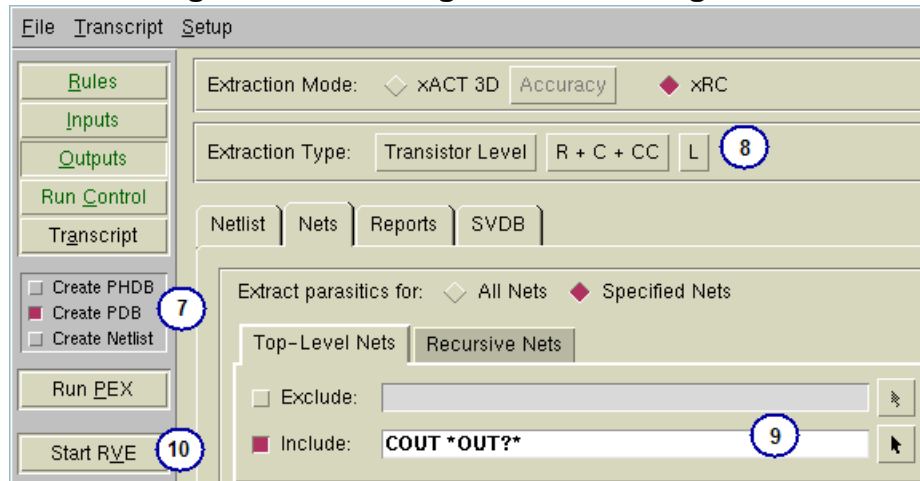
Figure 6-24. Generating PDB Incrementally



- a. Click the **Outputs** button in the left pane.
- b. Click the **SVDB** tab.
- c. Select **Generate PDB incrementally**.
6. After setting up the extraction type and any other settings, run extraction on the input.

7. In the PEX step controls, uncheck **Create PHDB**.
8. Reset the extraction type.

Figure 6-25. Adding Nets to Existing PDB



9. Specify the nets the new extraction type applies to under the **Nets** tab.
10. Click **Run PEX**.
11. Repeat steps 8 through 10 as needed.
12. Generate the netlist.
 - a. In the PEX step controls, select **Create Netlist**.
 - b. Uncheck **Create PDB**.
 - c. Click **Run PEX**.

Results

Check the Transcripts pane to verify the run completed with no errors. If you have selected “View netlist after PEX finishes” in the Outputs pane, a text viewer appears with the generated netlist loaded.

Related Topics

[Calibre xL User’s Manual](#)

Netlisting a Design Without Parasitics

To validate Calibre xRC output you might create a netlist showing only the intentional devices. This can be done with or without the usual parasitic extraction step. The rule file still contains rules for parasitic extraction.

Creating an Ideal Netlist from the Command Line	86
Creating an Ideal Netlist from Calibre Interactive	87

Creating an Ideal Netlist from the Command Line

Netlists without parasitics can be created for any of the flows. The only required difference is in the formatter step, where **-simple** replaces the typical **-all** or **-c**.

Prerequisites

- A valid PEX rule file for this layout including a [PEX Netlist Simple](#) statement specifying the output format.
- Layout database that is LVS-clean.

For more information refer to “[Basic Extraction Methods](#)”.

Procedure

1. Build database of intentional devices.

To use source names, use Calibre nmLVS-H:

```
calibre -lvs -hier -spice $svdb_dir/top_cell.sp rules
```

To use layout names, use Calibre xRC:

```
calibre -xrc -phdb rules
```

2. Optionally, extract parasitic effects.
3. Generate the netlist.

```
calibre -xrc -fmt -simple rules
```

Results

Successful Calibre xRC transcripts conclude with a count of errors and warnings as shown.

```

CALIBRE xRC WARNING / ERROR Summary
-----
xRC Warnings  =  2
xRC Errors    =  0
=====
```

If there are no errors, the directory also contains a file by the name you specified in the PEX Netlist Simple statement.

Creating an Ideal Netlist from Calibre Interactive

This procedure assumes you have a working SVRF file and only specifies the settings particular to producing non-parasitic netlists.

Prerequisites

- A valid PEX rule file for this layout.
- Layout database that is LVS-clean.

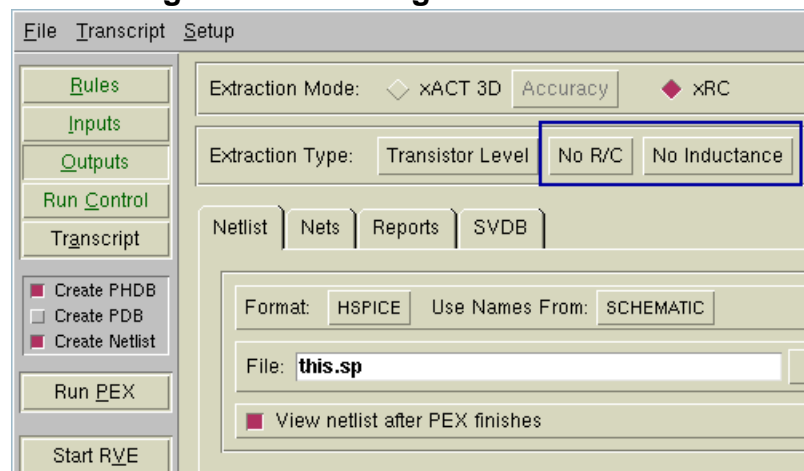
For more information refer to “[Prerequisites for Performing Parasitic Extraction](#)”.

Procedure

1. Start the PEX interface in Calibre Interactive.

```
calibre -gui -pex
```
2. Load a runset or rulefile.
3. Set the extraction type to extract no parasitics. (Any of the levels can be used; [Figure 6-26](#) happens to show transistor level.)

Figure 6-26. Setting for No Parasitics



4. Set other controls as needed. You can turn off the Create PDB step but it is not necessary.
5. Click **Run PEX** to produce the netlist.

Results

Check the Transcripts pane to verify the run completed with no errors. If you have selected “View netlist after PEX finishes” in the Outputs pane, a text viewer appears with the generated netlist loaded.

Backannotating Parasitics to a Source Netlist

To add layout parasitics to a source netlist, you must use the source-based flow. This calculates parasitic effects based on the layout, matches the layout devices to source devices, and attempts to map parasitics. If your source and layout are not close matches (for example, 2 source resistors are laid out as 24) you will get nodes labeled as “_noxref”.

Note



This method is not supported for full hierarchical extraction.

Backannotating from the Command Line.....	89
Backannotating from Calibre Interactive	90

Backannotating from the Command Line

This procedure describes how to back annotate parasitics from the command line.

Prerequisites

- A valid PEX rule file for this layout.
 - If you need to change pin order, model names, or parameters, use [PEX BA Mapfile](#).
 - The PEX Netlist statement should indicate SOURCEBASED.
- Layout database that is LVS-clean.

For more information refer to “[Prerequisites for Performing Parasitic Extraction](#)”.

Procedure

1. Build database of intentional devices. For backannotation, you must use Calibre nmLVS:

```
calibre -lvs -hier -spice $svdb_dir/top_cell.sp rules
```

2. Extract parasitic effects.

```
calibre -xrc -pdb -parasitic_switch rules
```

where **parasitic_switch** indicates the type of parasitics to extract. For example, -rc for distributed resistance and capacitance or -c for lumped capacitance.

3. Generate the netlist.

If **parasitic_switch** included resistance, use the following:

```
calibre -xrc -fmt -all rules
```

To produce a netlist that only contains lumped capacitance, use the following:

```
calibre -xrc -fmt -c rules
```

Results

Successful Calibre xRC transcripts conclude with a count of errors and warnings as shown.

```

CALIBRE xRC WARNING / ERROR Summary
-----
xRC Warnings   = 2
xRC Errors     = 0
=====
```

If there are no errors, the directory also contains the netlist you specified in the PEX Netlist statement. The exact number of files depends on the output format.

Backannotating from Calibre Interactive

The procedure described here is not specific to any layout viewer. There may be changes for your specific layout viewer or simulation flow.

Prerequisites

- A valid PEX rule file for this layout.
 - The [PEX Netlist](#) statement should indicate SOURCEBASED.
- Layout database that is LVS-clean.

For more information refer to [Basic Extraction Methods](#).

Procedure

1. Start the PEX interface in Calibre Interactive.
2. Load a runset or rulefile.
3. Specify the extraction type and other settings. Generally, for backannotation you use names from schematic.
4. Click **Run PEX** to produce the netlist.

Results

Check the Transcripts pane to verify the run completed with no errors. If you have selected “View netlist after PEX finishes” in the Outputs pane, a text viewer appears with the generated netlist loaded.

Related Topics

[Running Gate-Level Extraction](#)

Generating a Capacitance Summary Report

The capacitance summary report lists total capacitance and the ratio of coupling capacitance for all nets. It is controlled solely by the SVRF statement, PEX Report Netsummary. If the statement is present in the SVRF file, a run will produce the capacitance summary report regardless of other settings.

The report can be set up to provide details on only specific nets or cells. It can divide capacitance effects by cells, or report capacitance from top-level interconnect only.

Prerequisites

- A valid PEX rule file for this layout that includes a [PEX Report Netsummary](#) statement.
- Layout database that is LVS-clean.

For more information refer to “[Prerequisites for Performing Parasitic Extraction](#)”.

Procedure

1. Set up the PEX Report Netsummary information to provide the needed details.
2. Perform parasitic extraction for capacitance. (The extraction may also include resistance or induction effects.)

Results

Look for a file with the name specified in the PEX Report Netsummary statement. If no capacitance data was extracted, the report will end with “No meaningful analyzed data found.”

Related Topics

[Getting Started: Parasitic Extraction Using Calibre Batch Mode](#)

[Getting Started: Parasitic Extraction Using Calibre Interactive](#)

Generating a Net-to-Net Coupling Capacitance Report

The coupling capacitance report shows the amount of capacitance between specific pairs of nets and calculates how much this coupling capacitance represents of the total coupling on each net.

Reporting Coupled Capacitance from the Command Line	92
Reporting Coupled Capacitance from Calibre Interactive	93

Reporting Coupled Capacitance from the Command Line

Use this procedure to report coupled capacitance to a file from the command line.

Prerequisites

- A valid PEX rule file for this layout including a [PEX Report Coupling Capacitance](#) statement.
- Layout database that is LVS-clean.

For more information refer to “[Prerequisites for Performing Parasitic Extraction](#)”.

Procedure

1. Set up the PEX Report Coupling Capacitance information to provide the needed details.
2. Run parasitic extraction for coupled capacitance using either -c or -rcc for the parasitic flag.

Results

Successful Calibre xRC transcripts conclude with a count of errors and warnings as shown.

```
CALIBRE xRC WARNING / ERROR Summary
-----
xRC Warnings   =  2
xRC Errors     =  0
=====
```

If there are no errors and the extraction included coupled capacitance data, the directory also contains a file by the name you specified in the PEX Report Coupling Capacitance statement.

If the file is not present, check the transcript for warnings regarding PEX REPORT COUPLING CAPACITANCE.

Related Topics

[Standard Verification Rule Format \(SVRF\) Manual](#)

Getting Started: Parasitic Extraction Using Calibre Batch Mode

Reporting Coupled Capacitance from Calibre Interactive

This procedure includes only the minimum steps to generate the report. It can be created for any run that includes CC parasitics.

Prerequisites

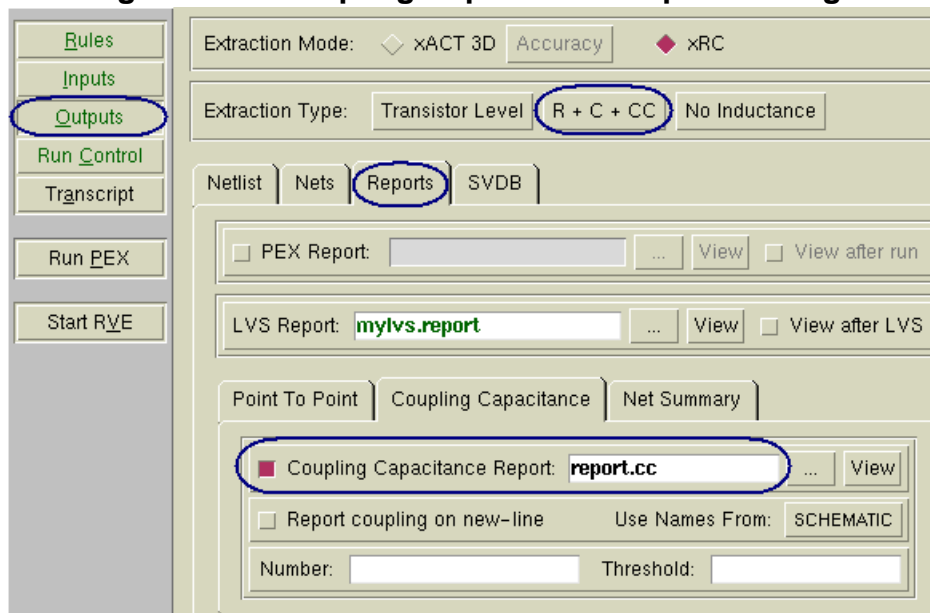
- A valid PEX rule file for this layout.
- Layout database that is LVS-clean.

For more information refer to “[Prerequisites for Performing Parasitic Extraction](#)”

Procedure

1. Start the PEX interface in Calibre Interactive.
2. Load a runset or rulefile.
3. In the Outputs pane, set the extraction type to **C + CC** or **R + C + CC**.
4. Under the **Reports** tab, enable the **Coupling Capacitance Report**.

Figure 6-27. Coupling Capacitance Report Settings



5. Provide a report name. (There is no default.) If needed, set the other options:
 - SPLIT_NET causes the pair of nets to be listed on two lines instead of one.

- Number sets the maximum number of nets to include in the report. Only the most tightly coupled pairs are reported.
 - Threshold sets the capacitance in farads below which not to report.
6. Set other controls as needed.
 7. Click **Run PEX** to produce the report (and netlist).

Results

Check the Transcripts pane to verify the run completed with no warnings about PEX Report Coupling Capacitance or errors.

If there are no errors, the directory contains the report file along with the netlist.

Related Topics

[Getting Started: Parasitic Extraction Using Calibre Interactive](#)

[Standard Verification Rule Format \(SVRF\) Manual](#)

Generating a Point-to-Point Resistance Report

Point-to-point resistance reports report on resistance along a single layout net. The points can be anywhere along the net. The point-to-point calculations take temperature effect into consideration when generating the report or looking at values in Calibre RVE. The Calibre Interactive interface allows you to turn the report on or off, but the contents of the report are controlled through a separate text file.

Note



It is easier to identify points if you label nets in the layout. Also, you will get more accurate results if you turn off PEX Reduce Ticer when creating this report.

Reporting Net Resistance from the Command Line.....	95
Reporting Net Resistance from Calibre Interactive.....	96

Reporting Net Resistance from the Command Line

This procedure describes the steps necessary to generate a net resistance report from the command line.

Prerequisites

- A valid PEX rule file for this layout including a [PEX Report Point2Point](#) statement.
- Layout database that is LVS-clean.

For more information refer to “[Prerequisites for Performing Parasitic Extraction](#)”.

Procedure

1. Set up the control file for the report. Each entry should be of the form

```
RESISTANCE Netname Location Netname Location
```

where

- *Netname* is the *layout* name of the net, and the same for both entries. (Resistance cannot be measured across devices.)
- *Location* is one of the following:

PIN <i>name signature</i>	Name of the resistor pin
PORT PROBE <i>p_name</i>	Name of port or probe
COORD <i>x y layer</i>	Coordinates and layer name to be used to find the closest node on the desired net. The coordinates are in database units.

The file can contain multiple entries. Each should be on a separate line.

2. Run parasitic extraction for resistance using any of the -r switches for the parasitic flag.

Results

Successful Calibre xRC transcripts conclude with a count of errors and warnings as shown.

```
CALIBRE xRC WARNING / ERROR Summary
-----
xRC Warnings   = 2
xRC Errors     = 0
=====
```

If there are no errors, the directory also contains a file by the name you specified in the PEX Report Point2Point statement.

Related Topics

[Standard Verification Rule Format \(SVRF\) Manual](#)

[Getting Started: Parasitic Extraction Using Calibre Batch Mode](#)

Reporting Net Resistance from Calibre Interactive

This procedure includes only the minimum steps to generate the report. It can be created for any run that includes parasitic resistance.

Prerequisites

- A valid PEX rule file for this layout.
- Layout database that is LVS-clean.

For more information refer to “[Prerequisites for Performing Parasitic Extraction](#)”.

Procedure

1. Set up the control file for the report. Each entry should be of the form

```
RESISTANCE Netname Location Netname Location
```

where

- *Netname* is the *layout* name of the net, and the same for both entries. (Resistance cannot be measured across devices.)
- *Location* is one of the following:

PIN <i>name signature</i>	Name of the resistor pin
PORT PROBE <i>p_name</i>	Name of port or probe

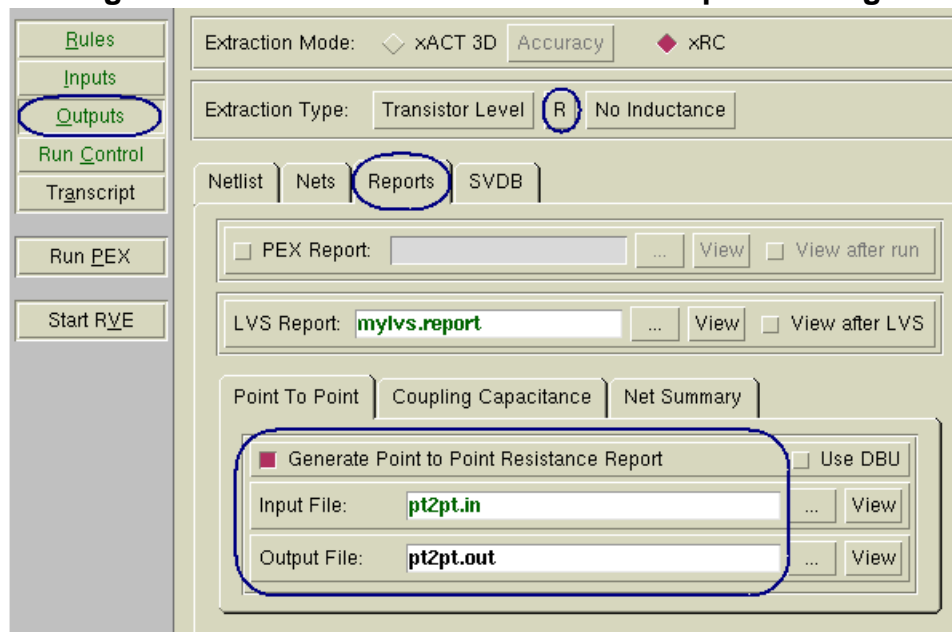
COORD $x y$ layer

Coordinates and layer name to be used to find the closest node on the desired net. The coordinates are in database units.

The file can contain multiple entries. Each should be on a separate line. The [PEX Report Point2Point](#) statement in the *Standard Verification Rule Format (SVRF) Manual* has more information.

2. Start the PEX interface in Calibre Interactive.
3. Load a runset or rulefile.
4. In the Outputs pane, set the extraction type to **R**, **R + C**, or **R + C + CC**.
5. Under the **Reports** tab, enable the **Generate Point to Point Resistance Report**.

Figure 6-28. Point-to-Point Resistance Report Settings



6. Enter the name of the control file you created in step 1 in the **Input File** field. Enter another name in the **Output File** field.
7. Set other controls as needed.
8. Click **Run PEX** to produce the report (and netlist).

Results

Check the Transcripts pane to verify the run completed with no errors. The directory contains the report file along with the netlist. If an entry in the report says “No analyzed resistors on net” the points in the entry have insignificant resistance or the net was not extracted.

Related Topics

[Getting Started: Parasitic Extraction Using Calibre Interactive](#)

[Standard Verification Rule Format \(SVRF\) Manual](#)

Extracting a Placed Cell

To extract individual cells, run in-context extraction. Only the specified cell instances are extracted, and not the top-level design.

In-Context extraction cannot be performed using the Calibre Interactive interface.

Prerequisites

- A valid PEX rule file for this layout.
- Hcell file or Hcell statement that includes all cells also listed in the xcell file.
- Layout database that is LVS-clean.

For more information refer to “[Prerequisites for Performing Parasitic Extraction](#)”.

Procedure

1. Identify the cell(s) in the xcell file using the following format:

```
cellname      -C parent_id/cell_id
```

The xcell file may contain entries for other cells, which will be treated as primitives. For instructions on finding layout IDs, see “[Discovering Layout Paths for In-Context Cells](#)”.

2. Build the database of intentional devices.

To use source names, use Calibre nmLVS-H:

```
calibre -lvs -hier -hcell hcell_file -spice $svdb_dir/top_cell.sp rules
```

To use layout names, use Calibre xRC:

```
calibre -xrc -phdb -hcell hcell_file rules
```

3. Extract parasitic effects for the cell.

```
calibre -xrc -pdb -xcell xcell_file -incontext -parasitic_switch rules
```

where *parasitic_switch* indicates the type of parasitics to extract. For example, -rc for distributed resistance and capacitance or -c for lumped capacitance.

4. Generate the netlist. Unlike with gate-level or full hierarchical extraction, the xcell file must be provided during netlisting.

```
calibre -xrc -fmt -xcell xcell_file -incontext -all rules
```

Results

Successful Calibre xRC transcripts conclude with a count of errors and warnings as shown.

```
CALIBRE xRC WARNING / ERROR Summary
-----
xRC Warnings  =  2
xRC Errors    =  0
=====
```

If there are no errors, the directory also contains netlists for each contextual xcell. The name of the netlist is *cellname.fmt*, where *cellname* is the name in the xcell list and *fmt* is the format specified in the PEX Netlist statement.

Related Topics

[Getting Started: Parasitic Extraction Using Calibre Batch Mode](#)

[In-Context Extraction](#)

Extracting Only the Top Level

To extract only top-level interconnect, place all cells instantiated in the top level in the xcell file. Use the cell names without IDs. Then run extraction using gate-level extraction. This occurs automatically with LEF/DEF layouts.

See [Running Gate-Level Extraction](#) for steps.

Extracting a Block Using CB

The Calibre® xRC™ CB™ tool performs parasitic extraction on designs containing up to 100,000 transistors depending on design complexity and computer memory.

Note



CB is not supported with hierarchical extraction.

Running Calibre xRC CB from the Command Line	100
Running Calibre xRC CB from Calibre Interactive	101

Running Calibre xRC CB from the Command Line

This procedure describes how to run Calibre xRC CB from the command line.

Prerequisites

- A valid PEX rule file for this layout, including the [Mask SVDB Directory](#) SVRF statement with any of the CCI, Query, or XDB keywords.
- A Calibre xRC CB license. See “[Licensing: Parasitic Extraction Products](#)” in the *Calibre Administrator’s Guide* for details.

Note



The Calibre xRC CB tool is a separately licensed parasitic extraction and netlisting product—for licensing information, see the [Calibre Administrator’s Guide](#).

- Layout database that is LVS-clean.

For more information refer to “[Prerequisites for Performing Parasitic Extraction](#)”.

Procedure

1. Build database of intentional devices. If you are using Calibre xRC CB, you probably also want to use Calibre nmLVS-CB as in this example:

```
calibre -lvs -nl -cb rules
```

2. When you extract parasitic effects, run with the -cb switch.

```
calibre -xrc -pdb -parasitic_switch -cb rules
```

3. Generate the netlist.

```
calibre -xrc -fmt -cb rules
```

Results

Successful Calibre xRC transcripts conclude with a count of errors and warnings as shown.

```
CALIBRE xRC WARNING / ERROR Summary
-----
xRC Warnings  =  2
xRC Errors    =  0
=====
```

Related Topics

[Calibre Verification User’s Manual \[Calibre Verification User's Manual\]](#)

Running Calibre xRC CB from Calibre Interactive

This procedure assumes you have a working SVRF file and includes only the minimum steps to use the CB license.

Prerequisites

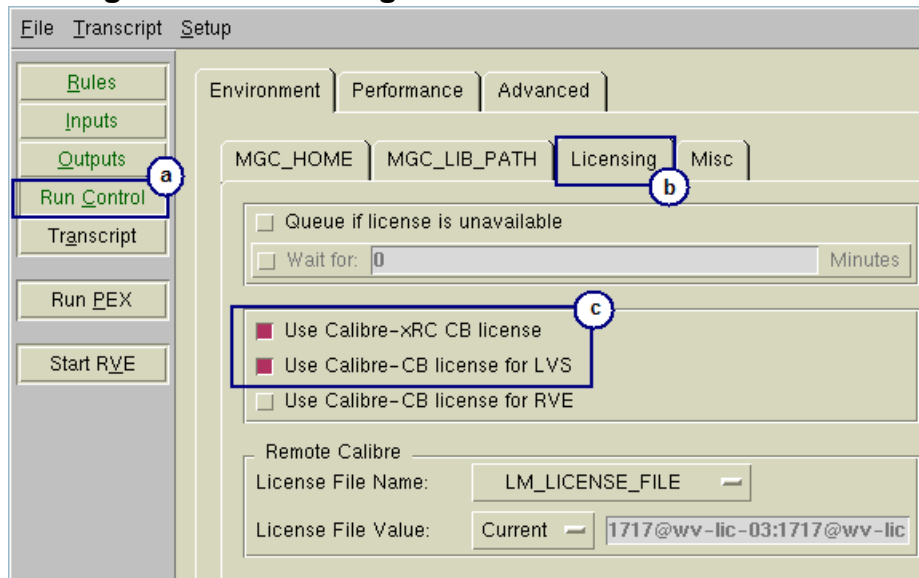
- A valid PEX rule file for this layout.
- A Calibre xRC CB license. See “[Licensing: Parasitic Extraction Products](#)” in the *Calibre Administrator’s Guide* for details.
- Layout database that is LVS-clean.

For more information refer to “[Prerequisites for Performing Parasitic Extraction](#)”.

Procedure

1. Start the PEX interface in Calibre Interactive.
2. Load a runset or rule file.
3. Enable CB license acquisition.
 - a. Click the **Run Control** button.

Figure 6-29. Enabling CB Runs in Calibre Interactive



- b. Select the **Licensing** tab.
- c. Enable the **Use Calibre-xRC CB license** and **Use Calibre-CB license for LVS** options.
4. Set other controls as needed. (The CB license cannot be used with hierarchical extraction.)
5. Click **Run PEX**.

Results

Check the Transcripts pane to verify the run completed with no errors. The directory contains the netlist. To verify that the CB license was used, check the transcript for the lines that begin with “Running:” These lines show the command line that was executed and will include a -cb switch when using the CB license.

Related Topics

[Calibre Verification User's Manual \[Calibre Verification User's Manual\]](#)

[Getting Started: Parasitic Extraction Using Calibre Interactive](#)

Chapter 7

Handling Input

Basic extraction processes can be modified to handle different types of input.

This chapter describes how to modify the basic extraction processes to handle the following types of variant input.

Hierarchy Control with Xcells	104
Calibre nmLVS-H and Calibre xRC Cell List Compatibility.....	105
Xcell List Format	105
Discovering Layout Paths for In-Context Cells	107
Wildcards in Xcell List.....	108
Tips For Choosing Xcells for Full Hierarchical Extraction	109
Metal Fill Modeling	109
Modeling Multiple Ground Regions	111
Varying Thickness with CMP Files	112

Hierarchy Control with Xcells

When performing hierarchical extraction, you define your design's hierarchy for Calibre xRC by identifying the lower-level cells in the design hierarchy.

You define this hierarchy by creating a cell correspondence list that matches source and layout names. This cell correspondence list, also known as an hcell list, is input into the Calibre nmLVS-H tool using the “-hcell” switch. An xcell list is derived from the hcell list; the xcell list can only contain cells that were named in the hcell list. The xcell list is specified as part of the Calibre xRC invocation.

Xcells may also be specified in your rule file using the [PEX Xcell](#) statement. You may use both methods as long as the xcell definitions in different locations are not in conflict. If a conflict is detected that cannot be resolved, then extraction stops with an error message. Two xcell flags or options will cause a conflict if they cannot be simultaneously applied to one xcell.

Conflicts between xcell specifications made in the SVRF rule file and the xcell list file can be resolved with the [PEX Xcell Precedence](#) statement. This statement allows you to control which file takes precedence if conflicting xcell definitions are encountered between files. If a conflict is detected, then a warning will be issued indicating the name of the affected cell and the xcell options that will be applied. If precedence has not been set and a conflict occurs, then extraction stops with an error. If conflicting xcell definitions are found within the same file, extraction will always stop with an error.

If all xcell specifications are made in your SVRF rule file, then the -xcell command line option for the PDB step is not required.

An xcell list is an ASCII file you create¹. The cells listed in the xcell list define your design's hierarchy and designate the cells the extraction will preserve. Cells not in the xcell list are flattened into the top circuit.

During Parasitic Database (PDB) creation, Calibre xRC caches a copy of the xcell list you input into the tool, even if the xcell list is empty. If you supply an empty xcell list, or if none of the cells in the provided xcell list are found in the layout, Calibre xRC issues the following warning message at the conclusion of the PDB stage:

```
WARNING: Could not match any layout cell names against the XCELL file:
        xcell_file_name
```

An empty xcell list effectively produces a transistor-level (flat) PDB and, consequently, a flat netlist or report.

1. As of version 2007.3, when the input is LEF/DEF format the hcell and xcell list are automatically created and used. You may customize the generated list but are not required to create it.

Calibre nmLVS-H and Calibre xRC Cell List Compatibility	105
Xcell List Format	105
Discovering Layout Paths for In-Context Cells	107
Wildcards in Xcell List	108
Tips For Choosing Xcells for Full Hierarchical Extraction.....	109

Calibre nmLVS-H and Calibre xRC Cell List Compatibility

When performing source name extraction, you must provide Calibre nmLVS-H with an hcell list that includes the cells you will use as xcells with Calibre xRC.

The hcell list uses the following format:

```
layout_name source_name
```

With Calibre nmLVS-H, you indicate the cell list using the “-hcell” switch. For example, the following command line invocations demonstrate a typical Calibre nmLVS-H and Calibre xRC run using the same cell list throughout, called *hcell_list*:

```
calibre -lvs -hier -spice svdb/design.sp -hcell hcell_list rules
calibre -xrc -pdb -rc -xcell hcell_list rules
calibre -xrc -fmt -all rules
```

The cell list performs the following functions in each of the tools:

- Calibre nmLVS-H — maps the layout’s cell names to their corresponding source’s cell names when you perform source name extraction. When invoking Calibre nmLVS-H, you specify the hcell list by using the “-hcell *hcell_list*” switch. See [calibre -lvs](#). See “[Hcells](#)” in the *Calibre Verification User’s Manual*.
- Calibre xRC — defines the design hierarchy for global net extraction. You create an xcell list based on the hierarchy you want, and Calibre xRC analyzes and extracts the hierarchy.

Although an hcell list can be used as an xcell list, xcell lists have additional options. If you add xcell options to your hcell list file, be sure to rename the file since these options do not work for Calibre nmLVS-H. Xcell entries can use wildcards to match multiple cells. Entries can also identify specific cell instances to extract “in context”.

Xcell List Format

An xcell list can have either of two formats, hcell or xcell.

You can use the hcell list format:

```
layout_name source_name
```

or you can use the xcell-specific format:

```
layout_name [source_name] [-C layout_path |  
-PCDEF [-INTRINSIC] |  
-P [-GDS file.gds] |  
-FILL file.gds cellname |  
-I ] [//Comment ]
```

In both formats, each cell is on its own line and can only appear once. If the xcell file follows the hcell list format, you can use the same file for both source name extraction and parasitic extraction. The second column (the source column) is ignored during parasitic extraction. The second, xcell-specific format can only be used for parasitic extraction and does not work as an hcell list for source name extraction.

The flags in the xcell-specific format can appear in either upper or lower case. [Table 7-1](#) provides more information on each flag.

Table 7-1. Xcell flags

Flag	Usage
-C <i>layout_path</i>	Used for in-context extraction as explained in In-Context Extraction . If doing gate-level extraction, the -C is ignored. See “ Discovering Layout Paths for In-Context Cells ” to determine a valid <i>layout_path</i> .
-PCDEF	<p>Indicates the cell is a pcell. The pcell contents are not extracted. The parasitics outside the pcell boundary will be extracted. For nets that exit an xcell through an xcell pin, the coupling capacitance between nets outside the xcell and nets inside the xcell will be extracted and netlisted connecting between the outside net and the xcell participating in the relationship. For nets totally enclosed within the xcell (they do not exit the xcell through any xcell pin), coupling capacitance is computed to the outside nets and is netlisted on the outside net dropped to ground.</p> <p>Can be specified with -I to output the intentional device information into the netlist.</p> <p>You can also use -PCDEF -I to generate an empty .subckt definition for the xcell, by specifying the xcell in an LVS Box statement and specifying LVS Preserve Box Cells YES in your rule file.</p>
-INTRINSIC	Used only in the PCDEF flow. Specifies that the pin intrinsic capacitance will be ignored for the device layers within the cell. Can only be specified in conjunction with -PCDEF.

Table 7-1. Xcell flags (cont.)

Flag	Usage
-P	Indicates the cell is a primitive. The contents are not extracted. In gate-level extraction, all cells are treated as primitives.
-GDS <i>file.gds</i>	Used only with LEF/DEF designs. Indicates that a cell is a GDS macro. Can only be specified in conjunction with -P.
-FILL <i>file.gds cellname</i>	Used only with the top cell of a LEF/DEF design. Specifies that metal fill is in a separate file, <i>file.gds</i> , in cell <i>cellname</i> .
-I	Indicates the cell is an ideal xcell. The contents are not extracted, but are written to the netlist. Only cells with the -I flag are treated as ideal xcells.

The following is a valid example of an xcell file:

```
// layout_name source_name flag
NOR      NOR      -I      //treated as an ideal cell
NAND     NAND     -P      //treated as a primitive
INV      INV      //handling depends on the extraction type
NMOS     NMOS     -PCDEF  //treated as a pcell
```

Note



The Calibre xRC tool disregards any [Hcell](#) statements you specify in the SVRF rule file. You must include any cell you identify with this statement in the xcell file or use the [PEX Xcell](#) statement.

Discovering Layout Paths for In-Context Cells

This procedure describes how to specify a valid layout path or “parent_id/cell_id” for -C cells in the xcell file.

Procedure

1. Create the PHDB using either method, source-name extraction:

```
calibre -lvs -hier -spice directory_path/filename.sp rule_file_name
```

or layout name extraction:

```
calibre -xrc -phdb rule_file_name
```

Layout name extraction does not create an XDB, which causes the Calibre Query Server to warn that the XDB is missing. You can ignore these warnings.

2. Start the Query Server by entering the following command in a terminal window. Specify your SVDB directory.

```
calibre -query svdb_directory
```

3. When the terminal shows “OK: Ready to serve.” enter the following command:

```
RESPONSE FILE query_results
```

The results will be written to the specified file in your current working directory instead of shown in the terminal. The server responds with “OK” when it is ready for the next command.

4. Write out the cells. You can either write out all cells, or just the entries for a few. To get a listing of all cells in the PHDB with their layout paths, enter the following:

```
PLACEMENT NAMES FLAT
```

To get the layout paths of specific cells, enter

```
PLACEMENTS OF cell_name FLAT
```

5. Exit the server by typing QUIT. The response file contains the valid paths for using in the xcell file.

Wildcards in Xcell List

You can use an asterisk (*) as a wildcard in the xcell list to ease the task of creating an xcell file.

When wildcards are used in xcell specifications, a cell name could match multiple xcell specifications. By default all matching xcell specifications are applied to the cell.

If the specifications use conflicting flags, Calibre xRC generates an error and extraction stops. For example, the following wildcard xcell specifications generate an error since a primitive xcell cannot be extracted in context:

```
pmos_rf* -I -P          // Treat as ideal xcell and primitive
pmos* -C x2/x1          // Extract in context
```

Use the [PEX Xcell Precedence](#) statement with the BEST keyword to resolve such conflicts, and select only the best matching wildcard xcell specification. The BEST keyword finds the best matching wildcard xcell name and applies only the flags for that specification. All other matches are ignored. With the BEST keyword, cells whose names match pmos_rf* would be treated as primitive xcells. The xcell specification to extract in context (-C) is ignored.

Caution


 Use the asterisk (*) wildcard in the xcell list with discretion as it could potentially increase runtime. It is recommended to specify as much of the name as possible before using a wildcard.

Table 7-2. Wildcards in an Xcell List

Xcell file without wildcards	Same Xcell file with wildcards
pmos_rf1	pmos_rf*
pmos_rf2	nmos_rf*
pmos_rf3	
pmos_rf7	
...	
nmos_rf1	
nmos_rf2	
nmos_rf3	
nmos_rf5	
...	

Tips For Choosing Xcells for Full Hierarchical Extraction

For most GDS-based extractions, you can shorten extraction time by using only a subset of the LVS hcells as xRC xcells. The cells you specify affect performance and accuracy.

- Choose cells that occur multiple times. The more times a cell appears, the faster extraction will be.
- Do not specify densely packed cells as xcells. In full hierarchical extraction, coupling between xcells is not modeled. For example, in a memory design, specify the cell containing an array rather than the cell containing a single bit.
- Do not specify cells that overlap another xcell. The contents in the overlapped area will be counted more than once.
- Do not specify cells with feedthrough nets as xcells, unless you are formatting the netlist in extended DSPF with the HSIM keyword set.

Metal Fill Modeling

By default, the Calibre xRC tool treats floating signal nets as fixed (grounded) and does not extract them. However, floating nets are not ignored when calculating the capacitance of non-floating nets.

Assuming that floating signals are grounded results in some error for floating nets like metal fill. You can control how metal fill, and floating nets in general, are treated during extraction with the [PEX Extract Floating Nets](#) statement. In the Calibre Interactive interface, this is accessed through **PEX Options > Netlist > Extract Metal Fill**.

To model the coupling resulting from metal fill, run parasitic extraction in -rc or -rcc mode and specify the SVRF statement as follows:

```
PEX EXTRACT FLOATING NETS ALL
```


This allows the metal fill to float, which gives a better approximation of real conditions. This may increase extraction time and the netlist size. If the netlists are too large to simulate, use [PEX Reduce CC](#) statement for targeted reduction.

For lumped capacitance extraction, which treats all neighbor nets as grounded, you need to extract the metal fill nets so that they can be simulated. To extract nets associated with the metal fill, set the statement as follows:

```
PEX EXTRACT FLOATING NETS GROUNDED
```

Floating nets and signal nets are extracted in the same manner.

Note

 If the metal fill has not been added yet, set the target density using the [PEX Density Estimate](#) statement. In-die tables must be included in the calibrated rule file for the process technology.

Many metal fill configurations are supported. [Figure 7-1](#) shows an example of simple square or rectangular fill between nets drawn on the same metal layer. [Figure 7-2](#) shows multi-layer fill and nets. [Figure 7-3](#) also shows a non-square fill with multiple nets.

Figure 7-1. Simple Metal Fill on a Single Layer

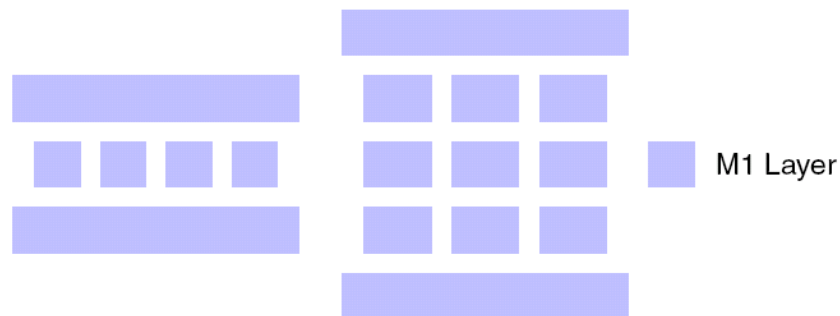


Figure 7-2. Metal Fill on Multiple Layers With Multiple Nets

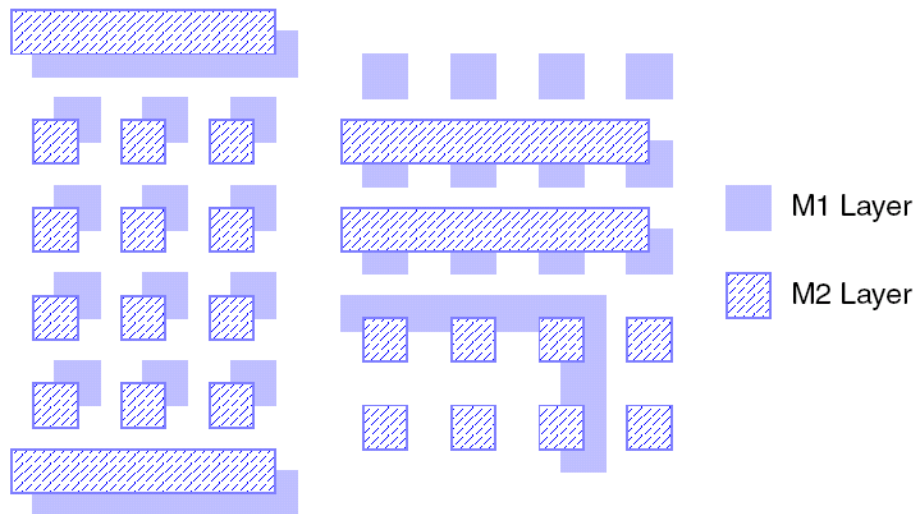
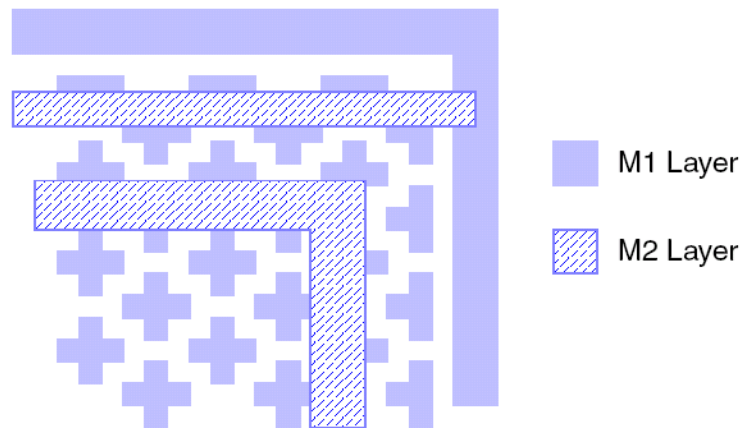


Figure 7-3. Non-Square Fill With Multiple Nets



For a more general discussion of floating nets, see [“Ignoring or Extracting Floating Nets”](#).

Modeling Multiple Ground Regions

Use this procedure when you need to reflect multiple ground regions of a design in your simulation, for example, when performing substrate noise analysis.

The steps result in ground regions “above” a global ground. When you extract, intrinsic capacitance goes to the named ground regions; grounded coupled capacitance goes to the global ground.

Note



You cannot model multiple ground regions with -rc extraction.

Prerequisites

- A design file with a separate layer for ground regions. Typically, this will be your p-well layer.
- A complete SVRF rule file including statements for calculating intrinsic capacitance. These are a normal part of the foundry-supplied calibrated rules.

Procedure

1. If it doesn't already exist, create a layer with shapes for the different ground regions in your layout. Different regions can be on different layers.
2. In the SVRF rule file, verify all ground layers have connectivity. (If you are using a well layer, it should already have connectivity.) The layer name must be in a [Connect](#) statement. For example:

```
CONNECT analog_regions
```

You can also connect related regions to each other with a [Virtual Connect Name](#) statement, for example:

```
VIRTUAL CONNECT NAME "digital_regions?"
```

3. Add [PEX Ground Layer](#) statement to the SVRF rulefile. For example:

```
PEX GROUND LAYER analog_regions digital_regions
```

4. To model ground regions not otherwise connected to a signal net, specify the [PEX Extract Floating Nets](#) statement in the rule file. You can select how the parasitic capacitance is calculated based on the GROUNDED, ALL, or REDUCED parameters for this statement.
5. Run extraction as usual. Distributed RC extraction and netlisting are not supported with multiple grounds, but all other types of parasitic models are. When running full hierarchical extraction, the ground regions are only reflected within the cells that contain the region-defining shapes.

Varying Thickness with CMP Files

If you have chemical mechanical polishing (CMP) models, such as CMPA, VCMP, or the database produced by Praesagus' Copper Prediction and Verification software, you can use its calculations for conductor thickness. The CMP data overrides all other methods of calculating thickness, such as PEX Thickness EQN or PEX Table. The Calibre xRC interpreter converts thickness values into the same units used by the extraction.

Because CMP data may not model all layers, you still must provide a process description by means of a technology file. When the CMP data and the process description both describe a layer, the CMP values are used.

Prerequisites

- Before you begin, you need a text version of the CMP data. For information on producing a text version of the database, see the documentation for the CMP modeling software.
- Make sure the layer names in the CMP data and the SVRF rule file are the same. Layers are matched by name and are not case sensitive. (For VCMP, the Calibre software treats all layer names as lowercase when looking for the corresponding file.)
- In order to use CMP data for conductor thickness, the extraction model must have a RHO_T or RSH_T table and a Thickness table for each conductor layer. For more information on the xCalibrate rule file generator and table syntax, refer to the [xCalibrate Batch User's Manual](#).

Procedure

1. Use the [PEX CMP Mode](#) statement in your extraction rule file to specify using CMP data during extraction.
2. Proceed with extraction using your usual methods. The CMP data can be used with all extraction modes.

Results

Calibre xRC uses the CMP thickness variation files for capacitance calculations and RHO-based or RSH-based resistance calculations in place of the models defined in the calibrated rule files.

If you get an error message about zero or negative thickness, check the following:

- Does the conductive layer have the same name (no misspellings) in both the SVRF file and the CMP data?
- Does the text file show a zero or negative value for the layer? In a Praesagus text file, the layer thickness is given as the last two values on a line that begins with the layer name.
- Is a layer used in the SVRF file missing in both the CMP data and the technology file?

If the run seems to be using the wrong values for layer thickness, check that all layer names are distinct regardless of case. Because the file parser does not flag duplicate names, when layer names differ only by capitalization the wrong layers may be matched.

Chapter 8

Tuning Extraction

The basic extraction flow can be modified to refine the extracted information.

This chapter describes how to modify the basic extraction procedure to refine the analysis and extraction phase as follows:

Extracting Devices without Parasitics	115
Extracting Particular Nets	116
Wildcards and Search Level Specification	117
Excluding Power and Ground Nets	118
Grounding Coupled Capacitors	118
Ignoring or Extracting Floating Nets.....	119
Extracting With Multiple Substrates.....	120
Resistance Extraction and PERC.....	123

Extracting Devices without Parasitics

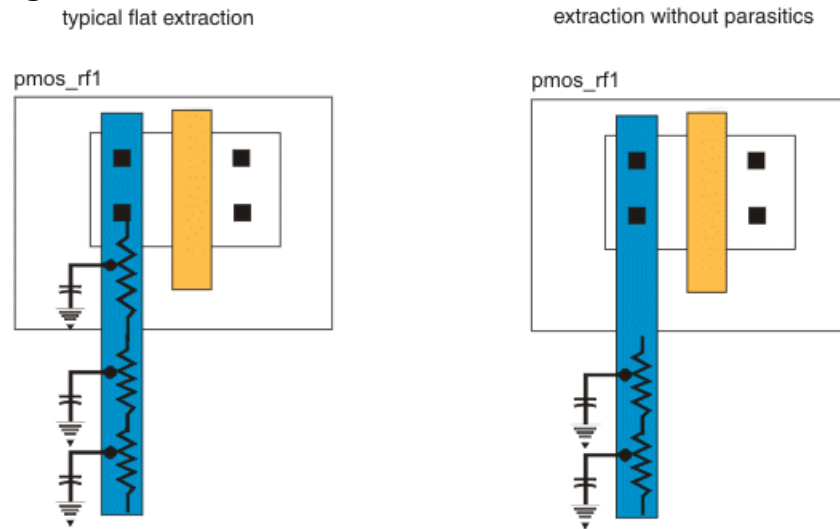
Calibre xRC supports extraction of devices that do not contain parasitics in the xcells. This is useful for flows where the device models already include the parasitics. For example, when physical layout of devices is done by using device generators or parameterized cells (pcells), and the device models are parameterized to match the layout.

To ensure accurate simulation results when using parameterized models, the extraction tool must not extract parasitics inside the specified devices. Anything in the design's xcell list is treated as ideal, meaning that no parasitics are included, and flattened to the transistor level. This method of extraction is sometimes known as “gray box” extraction.

Extracting devices without parasitics is not the same as extracting primitive xcells, because the netlist contains the nets for the xcell contents, which have been flattened within the xcell. All nets within the xcell are ideal nets, that is, they have no parasitic net models.

[Figure 8-1](#) compares a typical transistor level extraction with a pcell extracted without parasitics. Note that this method of extraction ignores only the parasitics inside the pcell. The parasitics outside the pcell are still extracted.

Figure 8-1. Device Extraction With and Without Parasitics



Invocation for Extraction Without Parasitics

To invoke extraction of devices without parasitic net models in the xcells (ideal cells), mark the cells with a -I switch in the xcell file.

Requirements

In order for this extraction method to provide useful information, the following requirements must be met:

- The gate-level option (-xcell *xcell_file*) for Calibre xRC must be set, even though the design is transistor level.
- Designs must use parameterized cells.
- The parameterized cell models must account for all of the parasitics in the cell.
- Each model must have an entry in the hcell file, and an identical entry in the xcell file. *All* entries will be treated as parameterized cells.

You can use an asterisk (*) as a wildcard in the xcell list, to ease the task of creating an xcell file, since pcells translated to GDSII have random numbers generated after the cell name. For more information on using wildcards with xcells, see [Wildcards in Xcell List](#).

Extracting Particular Nets

Use the following procedure to perform selected-net extraction with the Calibre xRC tool.

Procedure

1. Identify the selected nets by name using the [PEX Extract Include](#) statement in your SVRF rule file.

2. During parasitic database (*PDB*) creation, use the “-select” Calibre xRC command line switch. (Use -cselect to individually report coupled and intrinsic capacitance.)

If you omit the PEX Extract Include statement from your rule file and invoke the Calibre xRC tool using the “-select” switch, then the tool will issue an error and terminate the run.

For more information on -select and -cselect command line options, see “[calibre -xrc -pdb](#)” on page 157.

Wildcards and Search Level Specification

You can use wildcards to include or exclude nets when performing a parasitic extraction run. You can also specify where in the hierarchy to search for a matching net name.

- Use question mark (?) character as a wildcard matching zero or more characters in a net name.
- Use one of two mutually exclusive secondary keywords, TOPLEVEL and RECURSIVE, to specify where in the hierarchy to search for a matching name. TOPLEVEL is the default.

Example 8-1. Including Matching Names in Top Level

```
PEX EXTRACT INCLUDE LAYOUTNAMES TOPLEVEL "X0/X1/foo?"
```


[Example 8-1](#) includes any net in placement X0/X1 that is not ported out of that cell and has a name beginning with *foo*. For instance, this statement would include top level net names foobar, foos, and foo1, but would not include 1foo or ffoo.

Example 8-2. Including Matching Names From All Levels

```
PEX EXTRACT INCLUDE LAYOUTNAMES RECURSIVE "?in?" "?out?"
```

[Example 8-2](#) includes any source net, from any level of the hierarchy, that is not ported out and has the character strings *in* or *out* somewhere within the net name. For instance, this statement would include pin and pout, no matter what level of the hierarchy they occur on.

Note

 The tool supports wildcards in the net name only, not in the path. For example, *X0/X1/foo?* is supported, but *?/X0/vdd* is not supported. In other words, the wildcard character does not match hierarchy.

Excluding Power and Ground Nets

Excluding power and ground nets from extraction helps reduce runtime and parasitic netlist size.

Using the Calibre xRC tool, you exclude nets from the extraction run by including the [PEX Extract Exclude](#) statement in the rule file. When you use this statement, you must list the nets by name you want excluded from the extraction run.

Excluded power and ground nets are not removed from the layout database. Their coupling effect on signal nets is included in the parasitic netlist. However, the final netlist will exclude parasitics attached to these nets.

You disable net exclusion by commenting out the Exclude statement in your SVRF rule file.

You can use wildcards to select nets to exclude when performing a parasitic extraction run. You can also specify where in the hierarchy to search for a matching net name.

- Use question mark (?) character as a wildcard matching zero or more characters in a net name.
- Use one of two mutually exclusive secondary keywords, TOPLEVEL and RECURSIVE, to specify where in the hierarchy to search for a matching name. TOPLEVEL is the default.

Example 8-3. Excluding Matching Names in Top Level

```
PEX EXTRACT EXCLUDE LAYOUTNAMES TOPLEVEL "?vdd?"
```

[Example 8-3](#) excludes any name in the top level namespace that has the character string *vdd* anywhere within it. For instance, this statement would exclude top level net names *vdd*, *nvdd*, and *vdds*.

Example 8-4. Excluding Matching Names From All Levels

```
PEX EXTRACT EXCLUDE LAYOUTNAMES RECURSIVE "?vdd?"
```

[Example 8-4](#) excludes any net, at any level of the hierarchy, that is not ported out and has a name containing the character string *vdd*. For instance, this statement would exclude *nvdd* and *vdds*, no matter what level of the hierarchy they occur on.

Grounding Coupled Capacitors

You can force decoupling of coupled capacitors with the Calibre xRC formatter using the optional “-g” switch when you invoke the Calibre xRC formatter.

For example:

```
calibre -xrc -fmt -c -g rule_file_name
```

When you use this switch, Calibre xRC decouples coupled capacitors found in parasitic models, and grounds them.

Ignoring or Extracting Floating Nets

By default, floating nets are treated as grounded during extraction. A floating net is defined as one not connected to a device and lacking port text. (The shapes comprising the floating net may have text attached in other ways such as by Attach without Port Layer Text. Because this does not define port placement, connectivity is not defined and the net remains floating.)

Treating floating nets as grounded may affect the accuracy of coupled capacitance, usually by overestimating it. There are two methods for improving accuracy:

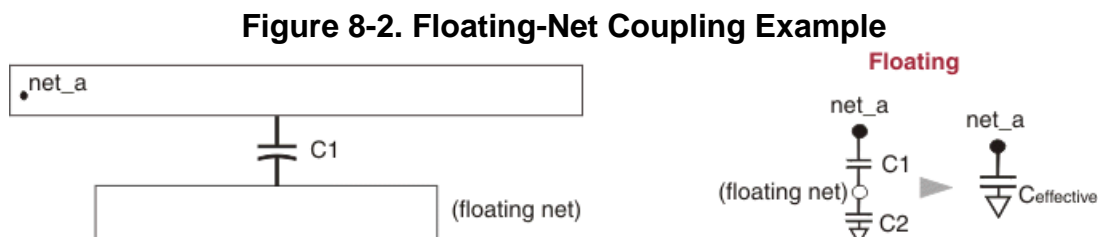
- [The Floating Net Coupling Algorithm](#)
- [Extracting Floating Nets](#)

Both are controlled through the [PEX Extract Floating Nets](#) statement, which is set through **PEX Options > Netlist > Extract Metal Fill** in Calibre Interactive.

The Floating Net Coupling Algorithm

Set the PEX Extract Floating Nets statement to REDUCED to more accurately extract capacitance in the presence of floating nets. By default, when the Calibre xRC tool performs RC extraction and converts coupling capacitances to grounded capacitances, it assumes floating signal nets are fixed (grounded) and does not extract them. This assumption results in some error for floating nets like metal fill.

When floating net coupling is enabled and a signal net is capacitively coupled to a floating net, the floating net is not assumed to be fixed. Instead, the Calibre xRC tool approximates the effective capacitance of the floating net and computes the effective (series) capacitance to ground of the signal net through the floating net as shown in [Figure 8-2](#).



This algorithm requires that there are signal nets and floating nets in the design, and is disabled if floating nets are extracted (that is, if [PEX Extract Floating Nets](#) is set to ALL). In contrast to extracting floating nets, the final netlist will not contain any floating nets.

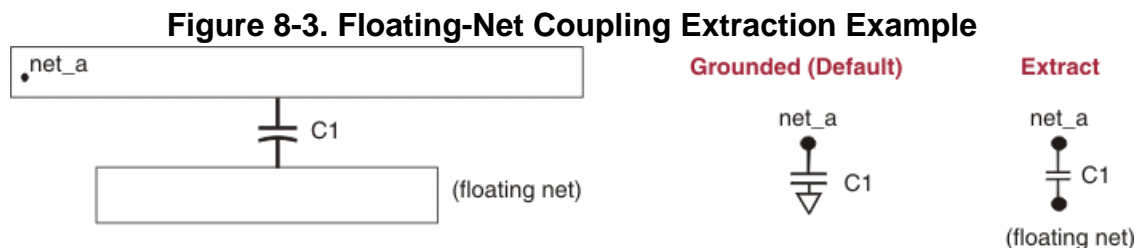
Note

When using selected nets (-select | -cselect switch), nets not selected are assumed to be grounded—this means that only selected nets can potentially float. To use the floating net coupling algorithm when running with selected nets, the floating nets must also be selected.

Extracting Floating Nets

Setting the PEX Extract Floating Nets statement to ALL causes Calibre xRC to output floating nets in the netlist and maintain the coupling between the signal and the floating net.

Figure 8-3 illustrates the Calibre xRC tool's result when you set this. It shows a test structure with a signal net, net_a, and unnamed floating net. C1 represents parasitic coupling capacitance. When you extract floating nets, C1 is represented as a coupling capacitor between net_a and the floating net rather than lumped with intrinsic capacitance. The netlist contains both nets, and C1 is listed with other coupling capacitors.



When extracting floating nets, both floating nets and signal nets are treated in the same manner. This increases extraction time and creates floating nets in the netlist.

Extracting With Multiple Substrates

In applications that integrate front-end RF devices in conventional CMOS technologies, the existence of high resistivity (low mobility) and low resistivity substrate areas within the same wafer is very common. Use this procedure to extract parasitics in the presence of multiple substrates.

This flow eliminates the need for separate calibration needed to produce the rule files for each substrate resistivity.

Prerequisites

- A design file with a multiple substrates. Typically, this is a situation where the substrates have different resistivity (mobility).
- Calibrated rule files containing multiple base layer definitions. For more information on defining multiple base layers, see “[Multiple Base Layers](#)” in *xCalibrate Batch User’s Manual*.
- All specified layers are connected.

- All regions built on very high resistivity substrate.
- A complete SVRF rule file including statements for calculating intrinsic capacitance. These are a normal part of the foundry-supplied calibrated rules.

Procedure

1. If it doesn't already exist, create a layer with shapes for the different ground regions in your layout. Different regions can be on different layers.
2. In the SVRF rule file, verify all ground layers have connectivity. (If you are using a well layer, it should already have connectivity.) The layer name must be in a [Connect](#) statement. For example:

```
CONNECT nwell
```

3. Verify that all substrate layers are connected.
4. Add [PEX Map](#) to map intrinsic models to ground layers. For example, base_High_R and base_Low_R are calibrated base layers defined in your .mipt file, and psub, nwell, and pwell are LVS layers.

```
PEX MAP base_High_R psub pwell
PEX MAP base_Low_R nwell
```

5. For processes that have overlapping wells that are at the same height, use [PEX Ground Layer](#) to define the order of precedence.

For mixed substrate region:

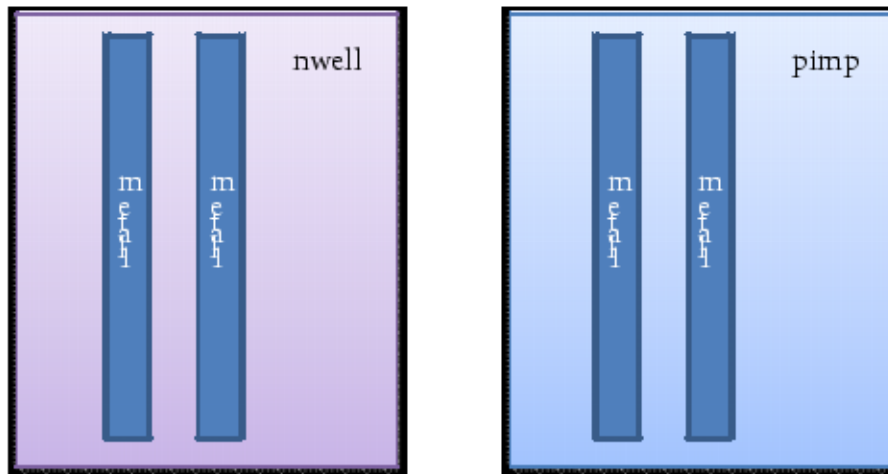
```
PEX GROUND LAYER nwell pwell psub
PEX MAP base_High_R psub pwell
PEX MAP base_Low_R nwell
```

6. Run extraction as usual. Distributed RC extraction and netlisting are not supported with multiple grounds, but all other types of parasitic models are. When running full hierarchical extraction, the ground regions are only reflected within the cells that contain the region-defining shapes.

Results

To validate that the capacitance values over different substrate areas behave differently, create a testcase with parallel lines running over two different substrate areas as shown in [Figure 8-4](#).

Figure 8-4. Parallel Line Example



With the same GDS file, run two tests:

LVS rule file:

```
layer diff 2
layer pimp 5
layer poly 6
layer cont 9
layer m1 10
layer vial 11
layer m2 12
```

Top level rule file:

```
// Example 1
// PEX MAP base_High_R diff
// PEX MAP base_Low_R pimp
// PEX NETLIST example1_xrc.dspf DSPF LAYOUTNAMES

// Example 2
PEX MAP base_Low_R pimp diff
PEX NETLIST example2_xrc.dspf DSPF LAYOUTNAMES
```

In Example 1, diff layer is mapped to the high resistance base and the pimp layer is mapped to the low resistance base. In the output netlist, the lines over the pimp region should have different capacitance values than the lines over the diff region. For example:

```
cg_1 diff1 0 0.1f
cg_2 diff2 0 0.1f
cg_4 pimp1 0 0.02f
cg_5 pimp2 0 0.02f
```

In Example 2, both the pimp and diff layers are mapped to the low resistance base. For the output of this example, the lines in both regions should have the same values. For example:

```
cg_1 diff1 0 0.02f
cg_2 diff2 0 0.02f
cg_4 pimp1 0 0.02f
cg_5 pimp2 0 0.02f
```

Resistance Extraction and PERC

You can set up a rule file for parasitic resistance extraction so that it can be used with Calibre® PERC.

For more information on how to do this, see “[Running Calibre PERC Using Calibre xRC Parasitic Resistance Netlist](#)” in the *Calibre PERC User’s Manual*.

Chapter 9

Controlling Netlisting

The basic extraction procedures can be customized in many ways to generate different types of netlist output.

These topics describe how netlisting behaves and how to modify some of the basic extraction procedures to produce different netlist output:

Netlisting Multiple User-Defined Corners	125
Netlisting Only Direct Devices on a Selected Net	126
Methods for Correcting Pin Swapping	127
Using the Source Based Flow	128
How to Join a Disjoint Parasitic Model	130
Net Name Formatting	131
Verification of Timing with Probe Points	131
Parasitic Extraction with Calibre View Device Properties	131

Netlisting Multiple User-Defined Corners

Follow this procedure when you want to netlist multiple user-defined corners in a single run. Process corners refer to the variations on a “typical” process; for instance, metal thickness may not be exactly controlled. Use this procedure for both foundry-supplied corners defined in a PEX Corner statement and variations on foundry-supplied corners that you define.

Prerequisites

- You must have a calibrated rule file containing capacitance and resistance rules that were created by xCalibrate version 2006.4_11 or later and that call `corner_index()`.
- The SVRF file must contain the [PEX Corner](#) statement and include the calibrated rule file.

Procedure

1. If you want to define your own custom corners, add a [PEX Corner Custom](#) statement to your SVRF file.
2. Run the PHDB and PDB steps as normal. The foundry-defined and user-defined corners are extracted.
3. Run the formatter with the `-corner` switch.

- To get all corners, use “-corner all”, for example:

```
calibre -xrc -fmt -corner all -all rules
```

- To specify particular corners, use the -corner switch with the corner names, for example:

```
calibre -xrc -fmt -corner typical,cc_worst -all rules
```

There should be no spaces around the comma separating two corners.

- If you do not specify particular corners, the typical corner is netlisted.
4. Corner netlists are named in the form *filename_corner*, where *filename* is specified in the PEX Netlist statement.

Netlisting Only Direct Devices on a Selected Net

To output a net, its devices, and the extracted parasitics without additional nets or devices, use the PRUNE keyword as part of the SVRF statement PEX Netlist and perform selected-net extraction.

Procedure

1. In the SVRF file, include these statements:
 - [PEX Extract Include](#) (to specify selected nets)
 - [PEX Netlist ... PRUNE ...](#) (to specify netlisting options)
2. During parasitic database (*PDB*) creation, use the “-select” Calibre xRC command line switch. For more information on the -select option, see “[calibre -xrc -pdb](#)” on page 157.
3. Run netlisting as usual.

Results

Only the selected net and its devices and parasitics will appear in the netlist. Although associated nets and devices are extracted and in the PDB, they are not written to the netlist.

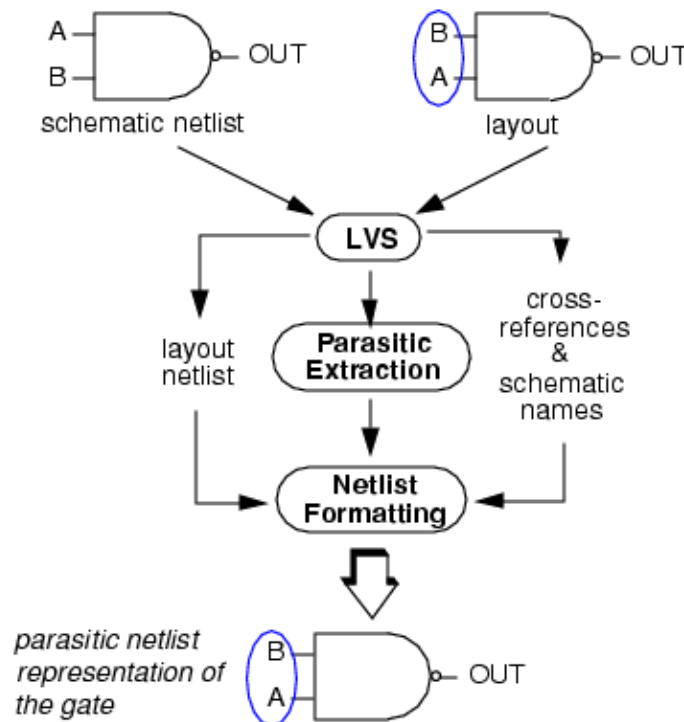
Methods for Correcting Pin Swapping

When performing parasitic extraction with the Calibre xRC tool, you can encounter issues with logical pin swapping in a distributed RC parasitic netlist.

The Calibre nmLVS-H tool performs logical pin swapping and propagates the swapped pins into the layout-to-source cross-reference files the Calibre xRC formatter uses when constructing a distributed RC netlist. Logical pin swapping at the gate level can create problems when using the netlist in a downstream simulation tool—the layout is LVS clean, but there is a pin mismatch between the schematic and the layout.

Figure 9-1 shows an example of logical pin swapping the Calibre xRC tool produces by default for a schematic NAND gate and its representation in the layout.

Figure 9-1. Gate-Level Logical Pin Swapping Example



In this example, the Calibre nmLVS-H tool reports the design is LVS clean and creates the layout-to-source cross-reference files using the swapped pins. By default, the Calibre xRC tool subsequently uses the swapped pins when creating the distributed RC netlist.

You can prevent pin ordering anomalies by using DSPF format. This format bases pin order on the pin names instead of the pin connections. If there are port direction restrictions in a certain output format, the formatter makes the appropriate adjustment. In SPEF, the formatter uses “b” for type “x” pins because SPEF grammar does not include an any-direction notation.

There are four methods to correct pin order. The one to use depends on whether you are using source-based flow, in which nets are based on source netlists, or the default layout-based flow (using layout names or schematic/source names).

- If you are using source-based flow — [Using the Source Based Flow](#)
- If the pins are on intentional models — [PEX BA Mapfile](#)
- If you are using a layout-based flow and the pins are on primitives or the top circuit only — [PEX Pin Order](#)
- If you need to specify pin direction or pin order on intermediate circuits — [How to Join a Disjoint Parasitic Model](#)

Using the Source Based Flow

Normally, the Calibre xRC formatter produces a layout netlist containing source names. The formatter uses the circuit pin order and cell hierarchy in the source netlist instead of the layout netlist when generating the extracted netlist.

When you use this flow, the Calibre xRC formatter produces a source netlist containing the extracted parasitics. The source-based flow is not supported for *full hierarchical extraction*.

Use the Calibre xRC formatter's source-based flow to:

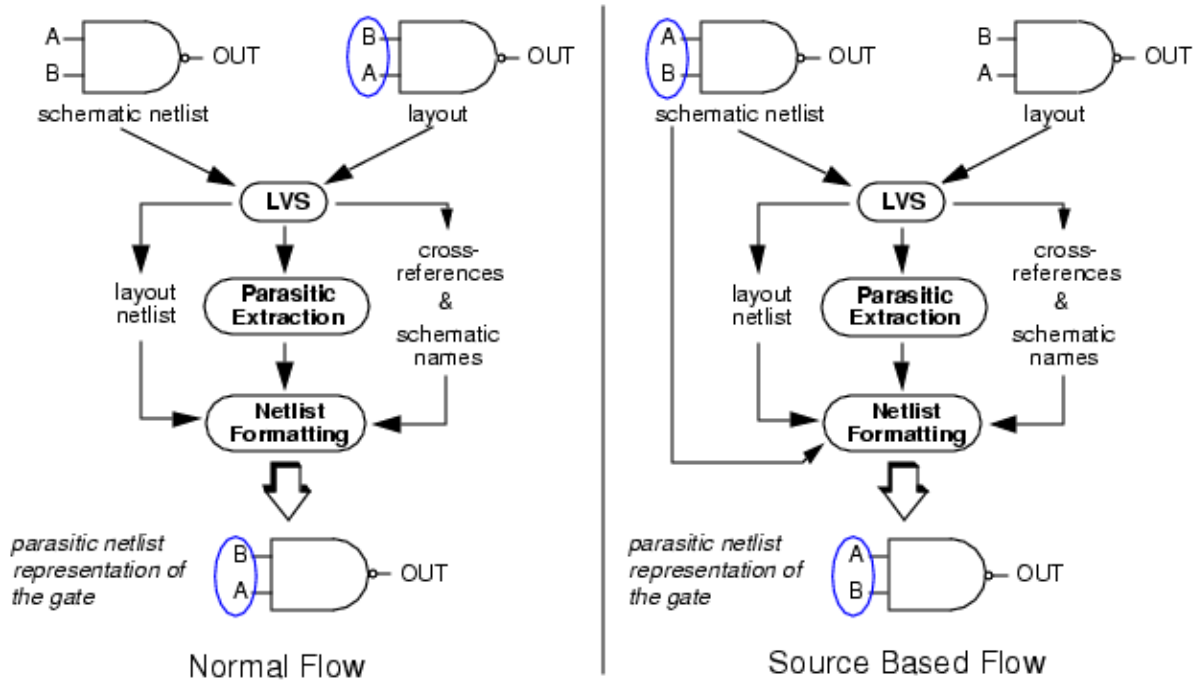
- correct logical pin swapping in the parasitic netlist.
- collapse multi-fingered devices back into one device.

If you are creating the source netlist with the Calibre nmLVS software, you must use Calibre nmLVS-H and the design must pass LVS. Because source-based extraction uses the schematic for formatting, it is very important that the layout pass LVS checks with no errors. If it is not, your formatting stage quits with the following error:

```
ERROR: Export name mismatch caused fatal error.  
Please verify that this design is LVS CLEAN.
```

[Figure 9-2](#) provides a comparison of the normal and the source based Calibre xRC formatter flows. When creating a parasitic netlist with the source based flow, the formatter also uses the schematic (source) netlist.

Figure 9-2. Comparison of Normal and Source Based Flows



Prerequisites

Before using this source based flow, you must modify your Standard Verification Rule Format (SVRF) rule file by adding or modifying the following SVRF statements:

- [Mask SVDB Directory](#) BY GATE
- [Source Case](#) YES

Note

When using the source-based flow, the Calibre xRC formatter automatically sets the Source Case SVRF statement to YES regardless of its setting in your SVRF rule file.

Procedure

1. To output each multi-fingered transistor as a separate device using the source-based flow, specify the [PEX Netlist](#) statement with the SOURCENAMES keyword in your rule file.
2. To collapse multi-fingered devices back into one device using the source-based flow, specify the [PEX Netlist](#) statement with the SOURCEBASED keyword in your rule file.
3. Generate the netlist.

Results

You can verify whether the Calibre xRC formatter used the source-base flowed by comparing the output netlist to the original LVS source netlist and seeing that the transistors are the same.

For example, given the original LVS source netlist sample:

```
MM7 net054 net37 vss! vss! N W=15u L=4u M=4
```

The Calibre xRC netlist output with PEX NETLIST SOURCENAMES specified is:

```
MM7 MM7:d MM7:g MM7:s MM7:b N L=4e-06 W=1.5e-05  
MM7@4 MM7@4:d MM7@4:g MM7@4:s MM7@4:b N L=4e-06 W=1.5e-05  
MM7@3 MM7@3:d MM7@3:g MM7@3:s MM7@3:b N L=4e-06 W=1.5e-05  
MM7@2 MM7@2:d MM7@2:g MM7@2:s MM7@2:b N L=4e-06 W=1.5e-05
```

Here the output netlist shows each multi-fingered transistor as a separate device.

The Calibre xRC netlist output with PEX NETLIST SOURCEBASED specified is:

```
MM7 MM7:d MM7:g MM7:s MM7:b N L=4e-06 W=1.5e-05 M=4
```

Here the output netlist shows the multi-fingered device as one device with a multiplicity of 4 (M=4), like in the original LVS source netlist.

How to Join a Disjoint Parasitic Model

Distributed parasitic extraction with Calibre xRC does not automatically connect multiple disjoint fragments of parasitic net models.

Use the [PEX Netlist Virtual Connect](#) statement to join a disjoint parasitic model. When you specify this statement with the YES parameter, the Calibre xRC formatter performs the following processing operations:

1. Looks for disjoint net model fragments created during LVS by Virtual Connect specification statements.
2. Subsequently connects these fragments to the “trunk” of the net model using a small-value resistor.

You can allow for incomplete net routing using the [Virtual Connect](#) SVRF specification statements during LVS and connect like-named net fragments. The Calibre xRC tool, however, will extract and produce a PDB containing a net model consisting of several disjoint parasitic model fragments — a true representation of the actual drawn design. Consequently, this will create problems for post-extraction simulation because there is no physical connection between these net fragments.

Note



You should use this statement in specific limited cases where you require a “clean” LVS when using Virtual Connect specification statements for locally disjoint power or signal nets. Extracting virtually connected nets often generates unexpected values, because the Calibre xRC tool arbitrarily connects virtually connected net segments when calculating extracted values. This could also cause problems with inductance extraction. Ultimately, your design should be LVS clean without using either Virtual Connect statements.

When you use the [PEX Netlist Virtual Connect](#) statement, the Calibre xRC formatter checks for disjoint net fragments. If the formatter finds such a fragment, the application selects a node on the fragment and connects this node using a low-value resistor to a node on the “trunk” of the net model. Although the formatter makes a reasonable choice of nodes when connecting, this connection is, essentially, arbitrary.

To create a listing of the connections the Calibre xRC formatter makes, use the `-fmt_warnings` switch on the command line during Calibre xRC formatter invocation.

Net Name Formatting

The Calibre xRC formatter uses the name of a connected port for the name of a net, even if the name of the port is different from the net name in the parasitic model. This is the standard behavior for Calibre xRC.

Verification of Timing with Probe Points

Within the parasitic model, a probe point exists as an internal connection with a unique *probe_name*. You can access each probe point by net and *probe_name*, depending on your simulator’s behavior.

The [PEX Probe File](#) statement allows you to verify timing from specific points on each net by setting up probe points. For more information on the PEX Probe File statement, refer to the [Standard Verification Rule Format \(SVRF\) Manual](#).

Parasitic Extraction with Calibre View Device Properties

Layout properties are calculated and represented in the Calibre View netlist produced by Calibre xRC.

Calibre xRC follows simple steps when calculating device properties. For details, see “[Device Property Calculation in Calibre View Generation](#)” in the *Calibre Interactive User’s Manual*.

Chapter 10

Integration and Troubleshooting Topics

This chapter contains information on tool integration and troubleshooting.

Integration	134
Guide to Calibre Interactive Files	134
Creating Batch Shell Scripts Using Calibre Interactive	135
Best Practices for Shell Scripts	137
Optimization	139
Improving Run Time	139
Creating Smaller Netlists	140
Best Way to Resize Designs	140
Troubleshooting	141
Setting Up For Troubleshooting	141
Invocation Issues	141

Integration

The goal of this section is to provide information for CAD engineers who are deploying pre-configured tools to their engineering groups.

Instructions for integrating the Calibre Interactive interface with layout viewers are available in the “[Setting the Calibre Interactive Socket Port for a Layout Viewer](#)” appendix of the *Calibre Interactive User’s Manual*.

Guide to Calibre Interactive Files	134
Creating Batch Shell Scripts Using Calibre Interactive.....	135
Best Practices for Shell Scripts.....	137

Guide to Calibre Interactive Files

The Calibre Interactive environment reads several types of configuration files. Settings in any of these affect how Calibre runs initiated from Calibre Interactive behave.

The configuration files are:

- Preferences file

The preferences file for Calibre xRC is *.cgipexdb*. You can point to a standard version using the `MGC_CALIBRE_PEX_RUNSET_FILE` environment variable. The default is *\$HOME/.cgipexdb*. Preferences files save the settings of the Setup Preferences dialog box and previously referenced runsets. When Calibre Interactive starts it automatically reads a preference file.

- Runsets

A runset is a text file created by Calibre Interactive to store the settings specified in the interface. Only non-default data is recorded. (Defaults can be changed by the preference file.) Users are typically prompted to load a runset at the start of each interactive session.

- Run scripts

A run script is any script which can be executed. For Calibre Interactive, run scripts are typically part of a trigger as described in the “[Trigger Functions in Calibre Interactive](#)” chapter of the *Calibre Interactive User’s Manual*. More generally, shell run scripts may also be used at the command line to execute a series of Calibre Interactive runs.

- Rule files

A rule file is a file that contains SVRF and TVF statements specifying the details of the Calibre run. Many SVRF specifications are only accessible in the rule file; a user cannot override them from Calibre Interactive.

Calibre Interactive loads files in the following sequence when invoked:

Table 10-1. Calibre Interactive Settings Sequence

Stage	Action
1	<p>Load the preferences file specified by \$MGC_CALIBRE_PEX_RUNSET_FILE. By default, this points to <i>\$HOME/.cgipexdb</i>.</p> <p>The preferences file populates the runset list and startup preferences.</p>
2	<p>Load a runset file and fill in the fields in the GUI.</p> <p>If the Load Runset File dialog has not been disabled by the preferences file, Calibre Interactive prompts the user to specify a runset. If the dialog has been disabled or the user cancels the dialog, no runset is loaded and default values are used in the fields.</p>
3	<p>Load files as user specifies.</p> <p>Typically, the user loads at least one rule file but may also load additional runsets. <i>Settings are not loaded into the GUI until the user clicks Load.</i></p> <p>Rule file and additional runsets can change fields already set in stages 1 and 2. The most recently loaded settings file (or manual setting by the user) is used at execution.</p>
4	<p>At execution, Calibre Interactive prepares a control file and runs any pre- and post-trigger run scripts.</p>
5	<p>At exit, Calibre Interactive prompts the user to save current settings to the loaded runset file. If the user made any changes, they will overwrite previous settings.</p>

In stage 4, Calibre Interactive prepares a control file. This file includes the rule file and GUI settings. Its name is based on the rule file name. For example, if the rule file is “*rules*”, then the control file is “*_rules_*”. This file is written to the working directory. Subsequent runs overwrite control files with the same name.

Creating Batch Shell Scripts Using Calibre Interactive

Many CAD engineers find it easiest to create new scripts by working in Calibre Interactive until they have a debugged set up, and then using Calibre Interactive’s files as the basis of a shell script which they deliver to their customers.

If the Calibre Interactive setup includes triggers, you will need to add additional calls to run the various executables. Because triggers are so flexible, this procedure does not attempt to describe transferring them to the shell script. You will also need to do additional programming if your solution uses customization files as described in the [Calibre Interactive User’s Manual](#).

Prerequisites

- A Calibre Interactive run that handles your requirements
- A text editor

Procedure

1. In Calibre Interactive - PEX, select **File > Control File > View**.

This opens a window with the control file displayed.

2. Transfer over the settings that were specified in the GUI. You can do this one of two ways:
 - Open your top rule file in a text editor. For each statement in the control file above `#IFDEF $MGC_CALIBRE_INTERACTIVE`, replace the line in the rule file with the corresponding control file version.
 - Use the control file as your new top rule file.
 - i. To avoid potential name collisions when the GUI writes control files, rename the control file to a different name that does not begin with an underscore.
 - ii. Set the environment variable `MGC_CALIBRE_INTERACTIVE` to 1 in your shell script.

The first method is better if you are concerned about editing SVRF files and attempting to override specifications. The second method causes the rule file to be interpreted as though the run were started from the Calibre Interactive GUI, and any overrides that appear in the top rule file above `DRC ICSTATION YES` are accepted.

3. In Calibre Interactive - PEX, select **Setup > Set Environment**.

This opens a window with the environment and shell variables you have used displayed.

4. In Calibre Interactive - PEX, click the **Transcript** button.

The transcript window shows the commands that will be executed when you click the **Run PEX** button.

5. Open your shell script in a text editor and make these changes:
 - a. Add any environment variables with a green check in the Runset column or under the **Shell Env.** tab to the variables section of your script.
 - b. Copy the `MGC_HOME` setting from the transcript to your script.
 - c. Copy the lines beginning “`$MGC_HOME/bin/calibre`” to the end of your script.
 - d. Add redirects at the end of each “`$MGC_HOME/bin/calibre`” line to create log files. (See [Example 10-1](#) in “[Best Practices for Shell Scripts](#)”.)

Results

Run the script from the command line and verify output.

Related Topics

[Best Practices for Shell Scripts](#)

[Guide to Calibre Interactive Files](#)

Best Practices for Shell Scripts

This section contains an example run script and tips for writing shell scripts. The tips are based on experience helping customers set up their solutions.

Table 10-2. Best Practices for Shell Scripts

Tip	Reason
Explicitly set the shell.	This ensures the script will run for all users, no matter their shell preference.
Use redirects to capture messages.	Many examples in training and on Support Center use “ tee” to capture logs as well as display messages in the terminal. However, “ tee” does not copy the standard error (stderr) output to the log file. Use “>&” or “>&!” in most shells to redirect both standard output and standard error to the same file.
Have the script remove any Calibre-generated files before invoking Calibre.	<p>Calibre leaves databases in the working directory so that users can come back to their results later. However, subsequent runs will overwrite data in existing files and this could cause misleading results — for example, left over netnames appearing in a netlist or erroneously swapped pins.</p> <p>Because some users may have a local version of rm, we recommend setting it explicitly also as</p> <pre>/bin/rm -rf</pre> <p>or as appropriate for your operating system.</p>
Set environment variables explicitly in the script rather than the user environment.	<p>Setting environment variables explicitly in the script serves several purposes:</p> <ul style="list-style-type: none"> • Easier to verify when debugging. • Reduces problems due to variables remaining set in the shell; they are only active while the script is executing. • Less typing than setting in the shell each time. • Easier to package complete testcases.

The following is an example shell script that follows best practices. Alternatives are provided together, with all but one commented out. This example run script can be used as the basis of your scripts.

Example 10-1. Example Shell Script to Run Calibre

```
#!/bin/csh -f

## Ensure which version will be run
setenv MGC_HOME /net/tools/calibre/2007.3_18.11/lv_micro.ixl
setenv PATH $MGC_HOME/bin:$PATH

## Clean the working directory
/bin/rm -r *svd* *SVD* lvs.log phdb.log pdb.log fmt.log *sum tr*

##### Use this environment variable for debugging #####
#####
#setenv CALIBRE_ECHO_RULE_FILE YES

##### Run-Specific Environment Variables #####
#####
#           Siemens EDA Variables
#setenv PEX_FMT_HP_PORT_MAP_MODE TEMPLATE
#setenv PEX_FMT_SOURCE_BASED_FLOW ON

#           Your Flow Variables
#setenv PROCESS_SIZE 90
#setenv PROCESS_TYPE CU
#setenv ADD_FILL NO

## Hcells for LVS and PHDB
set h = ""
#set h = "-hcell hcells"

## Xcells for xRC PDB
set x = ""
#set x = "-xcell xcells"

set r = "rules"
set c = "layout_primary"

##### Run PHDB, PDB, and FMT steps
$MGC_HOME/bin/calibre -lvs -hier -spice svdb/$c.sp $h $r >&! lvs.log
$MGC_HOME/bin/calibre -xrc -phdb $x $r >&! phdb.log
$MGC_HOME/bin/calibre -xrc -pdb -rcc $x $r >&! pdb.log
$MGC_HOME/bin/calibre -xrc -fmt -all $x $r >&! fmt.log
```

Optimization

The goal of this section is to provide tips for common optimization problems that CAD engineers are asked to tackle. Optimization is a broad topic, and these suggestions are by no means exhaustive.

Improving Run Time	139
Creating Smaller Netlists	140
Best Way to Resize Designs	140

Improving Run Time

There are several methods to make Calibre xRC runs shorter.

Not all of the following may apply to your situation.

- Run on multiple CPUs (-turbo).

By default, Calibre xRC consumes one license and runs on two CPUs no matter how many are available. Adding “-turbo” to the PDB stage causes the run to use as many CPUs as it has licenses for. For more information on this option, refer to [Calibre Administrator's Guide](#).

- Extract only the details you need.

Extraction can take a long time to run not just because of all the calculations but also because of the memory resources needed to process all the geometries of very large designs. Reducing the number of polygons by restricting input reduces both the number of calculations and the memory load.

- Use gate-level extraction to ignore devices.

Gate-level extraction uses an xcell file to indicate areas where parasitics are not extracted. Typically the contents of xcells are devices or standard cells, which already include parasitic effects in the simulation models.

- Exclude global nets such as power and ground.

Many types of analysis such as cross talk or static timing analysis only require data on signal nets. Parasitic effects on non-signal nets in these cases rarely modify simulation significantly. Both Calibre run time and simulation time will be sped up by omitting global nets in these cases.

- Use iterative extraction to get greater detail on only the critical nets.

Often the engineers only need fine detail on critical nets. These cases can run more quickly if the non-critical nets are extracted as resistance only (-r) or lumped capacitance only (-c). Then a select-net run can be performed on the critical nets with the required high level of detail as described in [Extracting Particular Nets](#).

Creating Smaller Netlists

There are two ways to create smaller netlists:

- Decrease the quantity of parasitic elements. This is known as reduction. Several reduction techniques are discussed in “[Reduction Techniques](#)” on page 199.
- Minimize the number of characters.

Generally, reduction techniques are preferred because they also reduce simulation times. If you need to use other methods to create more compact netlists, consider using the following SVRF statements:

[PEX Netlist Global Nets](#)

[PEX Netlist Noxref Net Names](#)

[PEX Netlist ... SPEF](#)

[PEX Netlist Create Smashed Device Names](#)

[PEX Netlist ... DSPF](#)

[PEX Netlist... HSPICE SHORTPINNAMES](#)

Best Way to Resize Designs

You can use Calibre to check whether layouts are suitable for process migration. Scaling designs and running DRC is relatively straightforward, but smaller sizes also require you to recalculate parasitic effects. (Note that below 90 nm, calibrated capacitance and resistance rule files older than v2008.1 are not recommended.)

When you are working with parasitics, you should scale your layouts using the [PEX Magnify](#) statement instead of the DRC methods of Magnify or Precision and Resolution because PEX Magnify can also scale your device properties.

If you are scaling the layout also for Calibre nmDRC and nmLVS, be sure that your combination of statements does not just change the scale. See “[Input Layout Database Magnification](#)” in the *Calibre Verification User’s Manual* for more details.

Troubleshooting

The goal of this section is to provide CAD engineers with some simple heuristics to aid in-house troubleshooting. It includes many of the steps used by Siemens support representative for initially diagnosing problems.

Setting Up For Troubleshooting	141
Invocation Issues	141

Setting Up For Troubleshooting

Follow these steps when you get a call from a user who is having trouble.

Prerequisites

- Working Calibre installation.
- Knowledge of shell commands for setting environment variables and creating log files.

Procedure

1. Get the full rule file by setting CALIBRE_ECHO_RULE_FILE to ON. This will echo out all included statements as well as the top rule file.
2. If you are using Calibre Interactive, be sure to also collect the control file, typically named “_rules_”.
3. If you are using scripts or entering commands directly at the prompt, be sure to redirect standard out and standard error to log files.
4. Be sure to use case-insensitive searching such as “grep -i” when looking through the log files. SVRF is generally not case sensitive, and Calibre messages may use mixed case.

Related Topics

[Best Practices for Shell Scripts](#)

Invocation Issues

If you are getting a usage message when you try to invoke Calibre, there is an error in the command line.

Check for these things:

- Are any of the switches incompatible? See “[Calibre xRC Tool Invocation Reference](#)” on page 151.
- Are there any errors in syntax such as missing arguments or switched order?

If it invokes but quickly exits, there should be an explanatory error message, such as “Could not get requested number of CPUs” or “problem with access of file”. If the run did not cleanly exit with an error message, and you can reproduce the problem, please open a [service request on Support Center](#).

Chapter 11

Handling Parasitic On-Chip Variation

Calibre xRC provides parasitic extraction techniques for on-chip variation analysis.

This chapter contains information on how to handle parasitic on-chip variation.

On-Chip Variation in Parasitic Extraction	143
Parasitic Extraction Techniques for On-Chip Variation.....	146
In-Die Variation	146
Process Corners.....	147
CMP Modeling	148

On-Chip Variation in Parasitic Extraction

On-chip variation refers to the inter- and intra-die variation in physical process properties. These variations can cause poor yields.

Layout data contains shapes that are at desired or “drawn” dimensions. The actual width of each line varies from the drawn dimension. This variation has both random and deterministic attributes. The random variations are caused by fluctuations in the manufacturing process. The deterministic variations are caused by the interaction of fabrication technologies and layout.

Sources of On-Chip Variation

There are several sources of on-chip variation:

- Fabrication equipment can vary. A wafer processed in an etch station will not be identical to a wafer processed in a neighboring etch station. This is sometimes referred to as “process variation”.
- Relative placements interact. The way lines and fill are placed (both spacing and internal width) changes how the actual drawn shapes appear. RET/OPC corrects as much of this as possible, but there is always some effect.
- Metals and dielectric can bulge or sag. Additionally, steps such as reactive ion etching cause some lines to be etched deeper than others. The changes in height are referred to as “loading”.
- The chemical-mechanical polishing (CMP) step will grind down the interiors of large polygons, and some regions of the wafer, more than others.

Modeling On-Chip Variation in Calibre Parasitic Extraction

The Calibre parasitic extraction software provides two techniques for modeling on-chip variation effects:

- In-die variation. Local density is used in conjunction with specialized rules to make adjustments to the drawn shapes. The adjusted shapes are used for calculating parasitic effects. Some foundries also supply loading effects in their in-die information. See “[In-Die Variation](#)” for more detail.
- CMP modeling. If your foundry provides appropriate CMP modeling files, you can have Calibre xRC read them in and make adjustments to metal thickness.

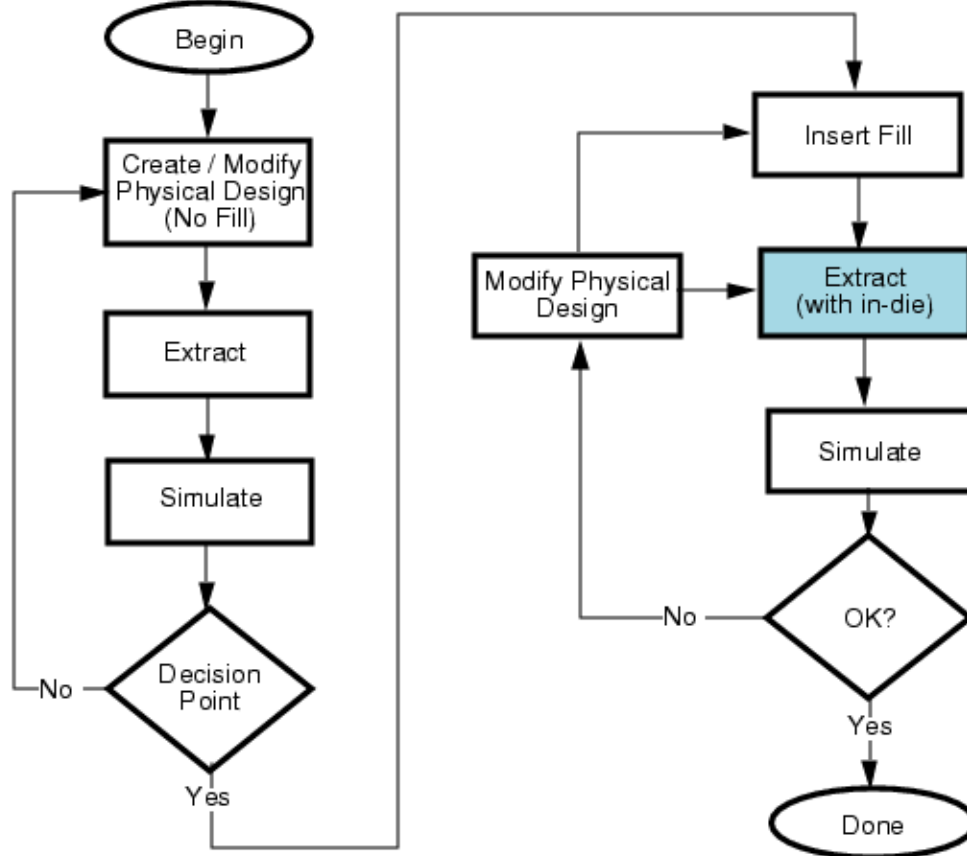
When to Use On-Chip Variation During Parasitic Extraction

The on-chip variation algorithms make small adjustments to parasitic extraction. They also increase the required amount of memory and can make the extraction step take longer to complete. They are recommended for use on near-final designs, as shown in [Figure 11-1](#).

If you haven't placed your metal fill yet, you almost certainly do not want to use in-die variation or CMP modeling. These are sensitive to local density, which will change when you add metal fill.

Process corners, which supply different nominal parameters such as temperature or resistance, can be run at any time with no penalty.

Figure 11-1. Design Flow Showing In-Die



Parasitic Extraction Techniques for On-Chip Variation

The parasitic extraction techniques for on-chip variation are independent of each other and may be singly or jointly used in extraction.

This section explains what the following techniques are doing and how to deploy them:

In-Die Variation	146
Process Corners	147
CMP Modeling	148

In-Die Variation

In-die variation is generally enabled by the foundry. The Calibre xRC user can turn it on or off, but not change its values.

Calibrated rules created before version 2008.2 allowed the use of manually created Parasitic Variation statements for similar effects, but the newer encrypted calibrated rules are not compatible with this.

Method

The following method is preferred, because the xCalibrate rule file generator can more accurately calculate the effect of in-die variation on the capacitance and resistance equations:


1. The foundry performs process measurements and determines how density and edge-to-edge distance affects properties such as edge displacement, thickness, temperature coefficients, and resistance.
2. The foundry creates in-die tables that follow the format described in “Table Syntax” in the *xCalibrate Batch User’s Manual*.
3. The foundry runs calibration, producing calibrated rules to provide as part of its design kit.
4. The CAD engineer at the chip design company provides wrapper scripts and SVRF files for the Calibre users. These may turn on or off in-die effects.
5. If the wrapper scripts do not turn on in-die functionality, the engineer running Calibre xRC may choose to enable in-die variation.

For manually-created calibrated rules, you can include in-die variation with the following technique:

1. Obtain information on in-die variation from the foundry.

2. Add **Parasitic Variation** statements to your rule file to model the effects of in-die variation on width, thickness, and resistance. Because you will not always want to include in-die effects, it is recommended to enclose the lines within a `#ifdef` preprocessor directive.

Note

 Older calibrated rule files may already include some Parasitic Variation statements.

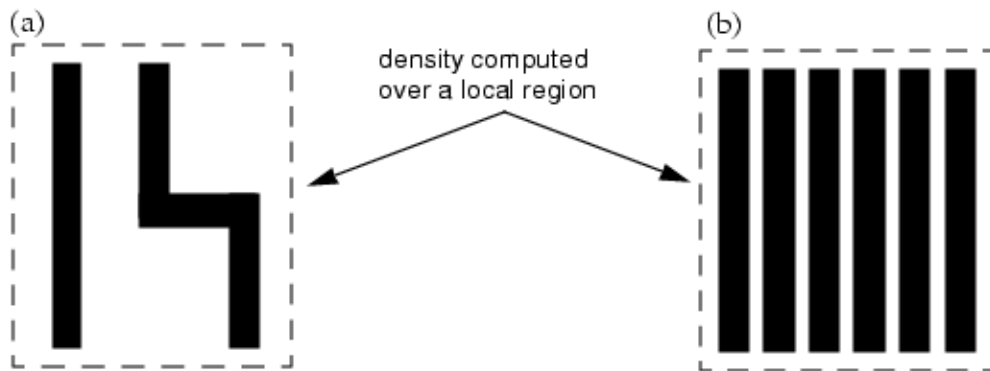
Theory

When appropriately calibrated, the Calibre xRC tool uses the drawn dimensions of each conductor along with the local density of the material in a region around the conductor to determine the actual width, spacing, and thickness of each line.

Key to this compensation is the idea of local density. The density of material in a region or window around each conductor affects the thickness of the line, and indirectly, the width.

Figure 11-2 shows how local density can vary from region to region.

Figure 11-2. Layout Structure Affects Local Density



In window (a), the area occupied by the layer material is relatively small compared to the area of the window itself; the local density is low. In window (b), the ratio of conductor area to window area is much higher, and local density is therefore much higher. Even though all of the lines in (a) and (b) have the same drawn width, the actual widths and thicknesses will differ between (a) and (b).

Process Corners

Process corners can be created by both the foundry and the chip development team. When Calibre xRC extracts parasitics, a process corner and all user-defined corners are extracted at once. You can choose to netlist all corners, or just a few.

When using ASIC optimizations, only one corner can be extracted at a time.

Method

1. The foundry sets up calibrations for multiple corners. The calibrated rules are provided as part of the process design kit.

The header of the calibrated rules includes a [PEX Corner](#) statement that lists the names of process corners available in the calibrated rules.

2. The design team determines which process corners need to be tested on designs before tape out, and adds any missing corners to the SVRF file using the [PEX Corner Custom](#) statement.
3. The engineer running parasitic extraction specifies the corners to produce using the -corner switch from a command-line invocation. (When using the Calibre Interactive interface, all corners are automatically extracted and netlisted.) See “[Netlisting Multiple User-Defined Corners](#)” for more information.

Theory

Process corners are a way of modeling manufacturing parameters. The minute differences between individual fabbers, even of the same model, result in different wafers receiving slightly more or less of a material than the typical amount. These differences can result in increased or decreased performance.

The CAD group at the semiconductor design company may want to verify process performance by running test simulations with more conservative parameters than the foundry felt necessary to include. The CAD group can specify particular parameters by layer that they want to change and have the engineers run verification with those.

The data for all corners is extracted during the PDB stage of a run. You cannot add a corner to a rule file after the PDB step and then produce output for it.

CMP Modeling

Most manufacturing processes use chemical-mechanical polishing (CMP) to planarize the wafer surface after deposition. The effect of the CMP step is calculated using complex software which creates data files. The foundry may supply those files as part of a design kit.

If you have CMP modeling files in the Praesagus or VCMP format, you can have Calibre xRC use them to modify thickness for the conductors. When used in conjunction with in-die variation, the CMP data will be used for modifying thickness instead of the in-die data.

Method

1. The foundry runs CMP modeling software and adds the data to the design kit.
2. The CAD group places the CMP files in a location accessible to the engineers running Calibre xRC.

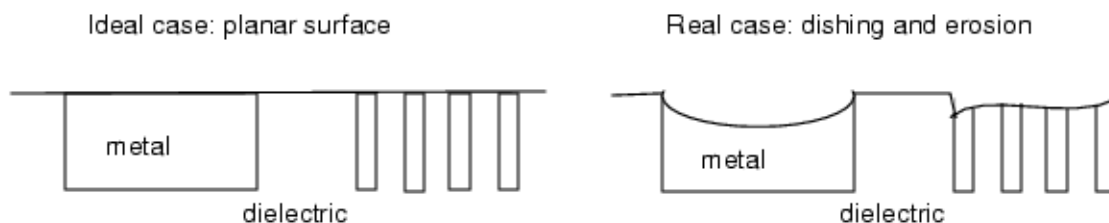
3. The CAD group sets up the environment variables and files for running CMP modeling. See “[Varying Thickness with CMP Files](#)” for more information.
4. When appropriate, the engineer running Calibre xRC enables CMP input during extraction.

CMP effects are sensitive to neighborhood density. Most of the time you would not benefit from including CMP data before placing metal fill.

Theory

CMP models predict how much erosion of the dielectric and dishing of metal occur on the wafer surface because of the CMP step in manufacturing. The models create a grid of the wafer and, for each square, determine the amount of polishing force or erosion. (The exact effect calculated by the CMP model depends on which modeling program you are using.)

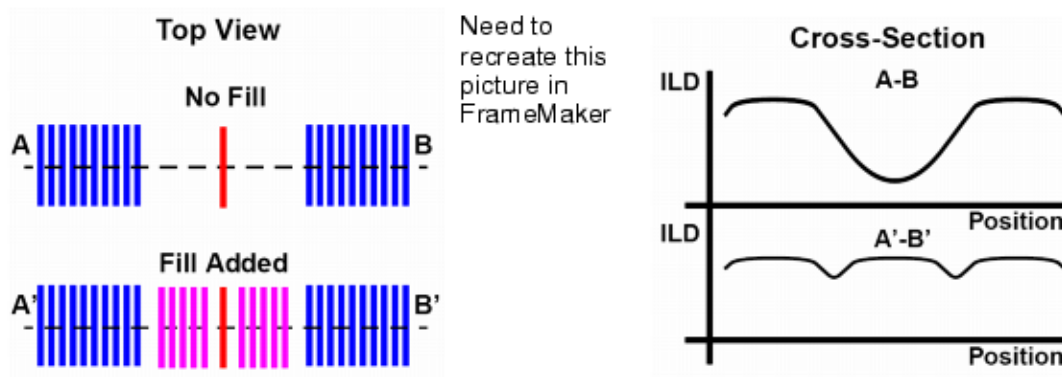
Figure 11-3. Typical Effects of CMP



[Figure 11-3](#) shows the potential effects of CMP. The metal is softer than the dielectric, and so is abraded more rapidly resulting in dishing. Where dense stretches of metal switch to dielectric, the polishing agent can “pile up” on one side, resulting in uneven erosion. Design geometry interacts with the CMP forces to dictate exactly how thickness will vary.

[Figure 11-4](#) illustrates a side effect of the interaction: metal fill placement completely changes the profile of the CMP erosion. Notice, however, that even very regular metal fill still does not eliminate the uneven wear.

Figure 11-4. Metal Fill and CMP



CMP models only predict how thickness will be affected. They can be used in conjunction with in-die variation, which also calculates density effects on edge placement. Because having a full view of neighboring shapes is important to both techniques, transistor-level (flat) extraction is best. This requires large amounts of disk space to complete the extraction, however.

Chapter 12

Calibre xRC Tool Invocation Reference

Calibre xRC command line options allow you to perform extraction from the command line or with a run script.

This chapter contains reference information for the invocation of the Calibre xRC tool.

Reference Syntax	151
Setting the CALIBRE_HOME Environment Variable	151
Command Invocation Reference	152
calibre -lvs.	153
calibre -xrc -phdb	154
calibre -xrc -pdb	157
calibre -xrc -fmt	160

Reference Syntax

The invocation descriptions use font properties and several meta-characters to document the command syntax.

For information on syntax conventions used in this chapter, refer to [Syntax Conventions](#).

Setting the CALIBRE_HOME Environment Variable

Calibre tools require that the CALIBRE_HOME environment variable be set.

See “[Setting the CALIBRE_HOME Environment Variable](#)” in the *Calibre Administrator’s Guide* for details.

Command Invocation Reference

Reference pages describe the command line invocation syntax for the Calibre xRC tool.

This section covers the following tool invocation commands:

- Source-Based connectivity and device recognition — [calibre -lvs](#)
- Layout-Based connectivity and device recognition — [calibre -xrc -phdb](#)
- Extraction and analysis of nets — [calibre -xrc -pdb](#)
- Format the netlist or report — [calibre -xrc -fmt](#)

calibre -lvs

Runs the Calibre nmLVS-H tool and creates the Persistent Hierarchical Database (PHDB) for use by the Calibre xRC tool.

Usage

calibre -lvs -hier -spice *directory_path/layout_primary.sp rule_file_name*

Description

When you invoke the Calibre nmLVS-H tool, you must specify an explicit path to the SVDB directory using the following syntax:


directory_path/layout_primary.sp

where

- *directory_path* is the path you specified in the [Mask SVDB Directory](#) statement in the rule file
- *layout_primary.sp* is the design's top-level cell you specified with the [Layout Primary](#) statement in the rule file

For complete Calibre nmLVS-H information, see the “[Calibre nmLVS and Calibre nmLVS-H Command Line](#)” section of the *Calibre Verification User's Manual*.

Note

 In standard LVS usage, you can specify any directory name for the SPICE netlist. To use the LVS output in the PDB stage, however, *directory_path* **must** match the Mask SVDB Directory setting.

Arguments

None.

Examples

The SVRF rule file is design.rules and contains the following SVRF statements:

```
LAYOUT PRIMARY "ring"  
MASK SVDB DIRECTORY "/scratch1/design/svdb" XRC
```

You would invoke the Calibre nmLVS-H tool from the command line using the following syntax:

```
calibre -lvs -hier -spice /scratch1/design/svdb/ring.sp design.rules
```

calibre -xrc -phdb

Performs connectivity extraction and device recognition, and writes the results to a PHDB.

Usage

calibre -xrc -phdb

```
[-cb | [-lbd @state_filename]  
[-turbo [number_of_cpus] [-turbo_all] [-remote host[,host...] | -remotefile filename]]]  
[-hcell hcell_list] [-E output] [-tvfarg argument] rule_file_name
```

Description

Performs connectivity extraction and device recognition, and writes the results to a [PHDB](#). This command also generates a layout netlist from a layout. After this step, you create the Parasitic Database ([PDB](#)) and format the netlist or report.

PHDBs must be re-generated if they are inconsistent with a current run; this can occur if you change the rule file.

Arguments

- **-xrc**
A required argument that invokes the Calibre xRC tool. For the PHDB stage, the netlists will contain layout names. Use [calibre -lvs](#) for schematic names.
- **-phdb**
A required argument that runs connectivity extraction and device recognition on the layout database specified in the rule file, and generates a PHDB.

The resulting PHDB is named *layout_primary.phdb*, where *layout_primary* is the name specified by the Layout Primary specification statement in the rule file.

This argument also generates a layout netlist named *layout_primary.sp* which is input to the formatter.

The Calibre xRC tool places both the PHDB and the layout netlist in the SVDB.
- **-cb**
An optional argument that runs connectivity extraction using xRC-CB licenses. This argument cannot be used with the **-turbo** or **-remote** arguments. See “[Extracting a Block Using CB](#)” for complete information.
- **-lbd @state_filename**
An optional argument that specifies that licenses reserved by the license broker daemon (LBD) be used during the PHDB creation stage. The *state_filename* argument is the name of the state file specified when starting the LBD.

For more information, see “[Calibre License Broker Daemon \(LBD\) for Calibre xRC](#)” in the [Calibre Administrator's Guide](#).

- **-turbo** [*number_of_cpus*]

An optional argument that specifies using multi-threaded parallel processing for PHDB creation. The *number_of_cpus* argument is a positive integer specifying the number of processors (CPUs) to use in the processing. If you omit this number, the Calibre xRC tool runs on the maximum available for which you have licenses. If you do not apply the -turbo argument, it defaults to running on two processors if available. To force the Calibre xRC tool to run on only one processor, specify “-turbo 1” on the command line.

For more information on this option, refer to [Calibre Administrator's Guide](#).

- **-turbo_all**

An optional argument used with the -turbo argument. This option halts the invocation if it cannot secure the exact number of CPUs specified using -turbo.

- **-remote** *host*[, *host*...]

An optional argument set that specifies to run the software on remote hosts using the MTflex multi-threaded, parallel processing architecture. It must be specified together with the -turbo option. It enables multi-threaded operation on remote hosts of a distributed network. You must specify at least one host parameter. A list of hosts is comma-delimited and specifies that multiple hosts participate in multi-threaded operations. You must have the required number of licenses for your job.

For more details, see the [Calibre Administrator's Guide](#).

- **-remotefile** *filename*

An optional argument set that is part of the MTflex multi-threaded, parallel processing architecture, which enables multi-threaded operation on remote hosts of a distributed network. It must be together with the -turbo option, which specifies the number of processors you are using, including those on the remote hosts. The *filename* specifies the pathname of a configuration file containing information for the local and remote hosts. You must have the required number of licenses for your job.

For more details, see the [Calibre Administrators Guide](#).

- **-hcell** *hcell_list*

An optional argument set that specifies the path to and name of the hcell file. It is used to create hierarchical PHDBs. See “[Hierarchy Control with Xcells](#)” for an in-depth discussion of hcells and xcells.

- **-E** *output*

An optional argument set that specifies an output file name for SVRF code generated by the TVF processor. If *rule_file_name* contains no TVF statements, *output* is empty. TVF code is processed before the run is started.

- **-tvfarg** *argument*

An optional argument set that specifies an *argument* that is passed to a compile-time TVF script. The *argument* can contain no space characters. The *argument* is read by the

tvf::get_tvf_arg command in the TVF rule file. For more information about TVF see the [Standard Verification Rule Format \(SVRF\) Manual](#).

- ***rule_file_name***

A required argument that specifies the path to and name of the SVRF rule file.

Examples

The following command invokes PHDB creation using a Calibre xRC license:

```
calibre -xrc -phdb -hcell hcells my_rules
```

The following command runs the same job on a computer named lsf3:

```
calibre -xrc -phdb -turbo -remote lsf3 -hcell hcells my_rules
```

The following command runs the same job on two remote host computers, lsf2 and lsf3:

```
calibre -xrc -phdb -turbo -remote lsf2,lsf3 -hcell hcells my_rules
```

The following command invokes PHDB creation using a Calibre xRC-CB license:

```
calibre -xrc -phdb -cb -hcell hcells my_rules
```

calibre -xrc -pdb

Extracts parasitic data and performs parasitic analysis on the data within the PHDB. It also generates the Parasitic Databases (PDBs) containing the electrical data for each net.

Usage

```
calibre -xrc -pdb [-r | -c | -rc | -rcc | -adms]  
  [-cb | [-lbd @state_filename] {[-turbo [number_of_cpus] [-turbo_all]]}]  
  [-xcell xcell_list [-incontext | -full]]  
  [-select | -cselect]  
  [-nocheck] [-pdb_info]  
  [-E output] [-tvfarg argument] rule_file_name
```

Arguments

- **-xrc**
A required argument that invokes the Calibre xRC tool.
- **-pdb**
A required argument that specifies to perform parasitic extraction and analysis on data in the PHDB, and generate the PDB.
- [-r | -c | -rc | -rcc | -adms] (choose one)
A required argument that selects the parasitic model. Choose one of the following:

-c	Specifies lumped and coupled capacitive parasitic extraction and places the capacitance models in the PDB.
-r	Specifies resistance-only extraction and places the R-only models in the PDB. Capacitance is not extracted.
-rc	Specifies distributed RC parasitic extraction and places the distributed RC models in the PDB. This mode calculates coupling capacitors but grounds them in the output netlist.
-rcc	Specifies R-coupled-C extraction and places fully-coupled models in the PDB.
-adms	Used in place of the -r, -c, -rc, and -rcc when running ADMS flow from the command line. See “Running ADMS Extraction” for the preferred method.
- **-cb**
An optional argument that runs connectivity extraction using xRC-CB licenses. See [“Extracting a Block Using CB”](#) for complete information.

- **-lbd @ *state_filename***

An optional argument set that specifies that licenses reserved by the license broker daemon (LBD) be used during the PDB creation stage. The *state_filename* argument is the name of the state file specified when starting the LBD.

For more information, see “[Calibre License Broker Daemon \(LBD\) for Calibre xRC](#)” in the [Calibre Administrator’s Guide](#).

- **-turbo *number_of_cpus***

An optional argument set that specifies using multi-threaded parallel processing for PDB creation. The *number_of_cpus* argument is a positive integer specifying the number of processors to use in the processing. If you omit this number, the Calibre xRC tool runs on the maximum available for which you have licenses. If you do not apply the -turbo argument, it defaults to running on two processors if available. To force the Calibre xRC tool to run on only one processor, specify “-turbo 1” on the command line.

For more information on this option, refer to [Calibre Administrator’s Guide](#).

- **-turbo_all**

An optional argument used with the -turbo switch. This argument halts the invocation if it cannot secure the exact number of CPUs specified using -turbo.

- **-xcell *xcell_list***

An optional argument set used to create the PDB hierarchically. This option specifies the path to and name of the file containing a list of cells to be preserved during extraction (xcells). For more information on xcells, see “[Hierarchy Control with Xcells](#)”.

When -xcell is specified without -incontext or -full, the primary cell is extracted to the boundaries of the xcells.

- **-incontext**

An optional argument used in conjunction with the -xcell argument set that extracts only those instances specified in the xcell list with a -C and a layout path.

- **-full**

An optional argument used in conjunction with the -xcell argument set. Performs hierarchical extraction with fully netlisted xcells. For more information, see “[Hierarchical Memory Extraction](#)”.

- **-select | -cselect**

An optional argument that specifies to exclusively extract nets using the net names you specify with PEX Extract Include SVRF statement. For more information, see “[Extracting Particular Nets](#)”.

The -cselect argument can only be used for lumped and coupled capacitive (-c) extraction and R-coupled-C (-rcc) extraction, and cannot be specified with -select. When extraction is run with -cselect, coupled capacitance is kept distinct from intrinsic capacitance; with -select, the two are lumped together.

- **-nocheck**
An optional argument that continues the extraction run with only a warning if file date stamps (commented checksums) are inconsistent with each other. If **-nocheck** is not specified and the date stamps are inconsistent, the extraction run stops.
- **-pdb_info**
An optional argument that controls whether THRESHOLDING messages from the analyzer are printed to the transcript. If you add the option to the command line, messages from the analyzer are printed to the transcript. If you do not use the option, messages from the analyzer are not printed to the transcript.
- **-E *output***
An optional argument set that specifies an output file name for SVRF code generated by the TVF processor. If ***rule_file_name*** contains no TVF statements, *output* is empty. TVF code is processed before the run is started.
- **-tvfarg *argument***
An optional argument set that specifies an *argument* that is passed to a compile-time TVF script. The *argument* can contain no space characters. The *argument* is read by the `tvf::get_tvf_arg` command in the TVF rule file. For more information about TVF see the [Standard Verification Rule Format \(SVRF\) Manual](#).
- ***rule_file_name***
A required argument that specifies the path to and name of the SVRF rule file.

Examples

This example extracts only those cell instances marked for extraction in the xcell file.

```
calibre -xrc -pdb -rc -xcell xcells -incontext my_rules
```

This example shows an iterative run.

```
calibre -xrc -phdb -hcell hcells my_rules
calibre -xrc -pdb -rc my_rules
calibre -xrc -pdb -rcc -select my_rules
calibre -xrc -fmt -all my_rules
```

calibre -xrc -fmt

Produces netlists and reports from the PDB contents.

Usage

```
calibre -xrc -fmt [-netmodel] [-c | -r | -rc | -rcc | -adms | -all | -simple]  
  [-g] [-cb | -lbd @state_filename]  
  [-xcell xcell_list [-full | -incontext]]  
  [-corner {corner[,corner]... | all}]  
  [-fmt_warnings] [-fmt_info]  
  [-nocheck]  
  [-E output] [-tvfarg argument] rule_file_name
```

Description

Produces netlists and reports from the PDB contents. Standard output formats are generated from:

- Parasitic models stored in PDB(s)
- Source-to-layout cross-reference files, if they exist
- SPICE layout netlist
- Rule file specification statements
- Specified environment variables

The supported output formats include HSPICE, DSPF, Spectre, Eldo, SPEF, CalibreView, and PrimeTime; and ASCII reports.

You specify the output formats and filename locations with the PEX Netlist, PEX Netlist ADMS, PEX Netlist Select, PEX Netlist Simple, and PEX Report statements in the rule file.

Arguments

- **-xrc**
A required argument that invokes the Calibre xRC tool.
- **-fmt**
A required argument that specifies to generate netlists and reports from parasitic data stored in the PDB.

This argument generates netlists and reports from parasitic data generated during selected and flat extraction. Selected nets and flattened global nets are processed with the same command line options because, in both cases, the parasitic model represents nets extracted in their entirety and flattened to the top-level cell. Netlists and reports use layout netlist names unless source names are specified in the rule file. Parasitic models are named *cell_name%net_name* in the output netlists.

- **-netmodel**

A required argument that specifies to perform netlist formatting using a file used for selected nets with assigned net models. The SVRF rule file must contain a [PEX Netlist Select File](#) statement. This option cannot be used with **-c**, **-r**, **-rc**, **-rcc**, **-adms**, or **-all** options. It can only be used with the **-simple** option.

- **{-c | -r | -rc | -rcc | -adms | -all | -simple}**

A required argument that selects the output mode. Choose one of the following:

- c** Exclusively writes capacitance models into netlist. Do not use when **-netmodel** is specified.
- r** Exclusively writes resistance models into netlist. Do not use when **-netmodel** is specified.
- rc** Distributed netlist; suppresses writing of any inductors; if coupled capacitance data exists, it is grounded; exits if no parasitic resistance data is in the PDB. Do not use when **-netmodel** is specified.
- rcc** Distributed netlist; suppresses writing of any inductors; exits if no parasitic resistance or coupled capacitance data is in the PDB. Do not use when **-netmodel** is specified.
- adms** ADVance MS-specific netlist. Used when running ADMS flow from the command line. See [“Running ADMS Extraction”](#) for the preferred method. Do not use when **-netmodel** is specified.
- all** Distributed netlist; writes the contents of the PDB into the netlist specified by the PEX Netlist statement. If the PDB contains only lumped capacitance data, the run will stop with an error. This switch is optional and is the default. Do not use when **-netmodel** is specified.
- simple** Produces a simple netlist with no parasitic elements.

- **-g**

An optional argument that grounds any coupling capacitors.

- **-cb**

An optional argument that runs connectivity extraction using xRC-CB licenses. For more information see [“Extracting a Block Using CB”](#).

- **-lbd @ *state_filename***

An optional argument set that specifies that licenses reserved by the license broker daemon (LBD) be used during the formatting stage. The *state_filename* argument is the name of the state file specified when starting the LBD.

For more information, see [“Calibre License Broker Daemon \(LBD\) for Calibre xRC”](#) in the [Calibre Administrator’s Guide](#).

- **-xcell *xcell_list***
An optional argument set formerly required with all hierarchical PDBs to specify the path to and name of the xcell file. This argument is retained for backwards compatibility in scripts.
- **-full**
An optional argument that specifies that the formatter should produce a fully hierarchical netlist.
- **-incontext**
An optional argument that tells the formatter to output only the contextualized cells, without the nets of the parent.
- **-corner {*corner[,corner]...* | all}**
An optional argument set that selects the process corners to write out. If you specify two or more corner names, do not place a space between names. If the rules define multiple corners and you do not use this option, only the typical corner is netlisted.

When **-corner** is used in the formatter stage, sensitivity netlisting is disabled and sensitivity variations do not appear in the netlist.
- **-fmt_warnings**
An optional argument that displays warning messages while the Calibre xRC formatter runs. Otherwise, messages are not displayed.
- **-fmt_info**
An optional argument that displays informational messages while the Calibre xRC formatter runs. Otherwise, messages are not displayed.
- **-nocheck**
An optional argument that continues the extraction run with only a warning if file date stamps (commented checksums) are inconsistent with each other. If **-nocheck** is not specified and the date stamps are inconsistent, the extraction run stops.
- **-E *output***
An optional argument set that specifies an output file name for SVRF code generated by the TVF processor. If **rule_file_name** contains no TVF statements, *output* is empty. TVF code is processed before the run is started.
- **-tvfarg *argument***
An optional argument set that specifies an *argument* that is passed to a compile-time TVF script. The *argument* can contain no space characters. The *argument* is read by the `tvf::get_tvf_arg` command in the TVF rule file. For more information about TVF, see the [Standard Verification Rule Format \(SVRF\) Manual](#).
- ***rule_file_name***
A required argument that specifies the path to and name of the SVRF rule file.

Examples

To write only capacitances to the output netlist, use the following invocation:

```
calibre -xrc -fmt -c -xcell xcells my_rules
```

When using a net file to define how specific nets should be formatted, use the following invocation:

```
calibre -xrc -fmt -netmodel my_rules
```

The *my_rules* file will specify the net file *my_selected_nets* as follows:

```
PEX NETLIST SELECT FILE my_selected_nets
```

For more information on [PEX Netlist Select File](#), see the *Standard Verification Rule Format (SVRF) Manual*.

Appendix A

Parasitic Effects and Calibre Tools

The information in the following sections focuses on the Calibre xRC parasitic extraction tool and the xCalibrate rule file generator.

Parasitic electrical effects are well understood and have complex differential equations to describe them. These equations are used by the category of tools called “field solvers”. Field solvers have the highest accuracy for predicting real-world consequences of layouts. However, field solvers also take a very long time to run and are generally unable to cope with more than a few dozen nets at a time.

The Calibre parasitic extraction tools abstract the interactions into a set of multi-variable polynomial equations. These equations compute parasitic effects far more quickly than field solvers.

Note



Typical chip structures do not include skew interconnect at angles other than 45 degrees, or conformal interconnect layers whose vertical dimensions overlap.

The parasitic effects and Calibre models are described in the following sections:

Parasitic Capacitors.....	165
Capacitance Models in Parasitic Extraction.....	166
Parasitic Resistors.....	169
Resistance Models in Parasitic Extraction.....	170

Parasitic Capacitors

Capacitance exists anytime you have two conducting layers in close proximity separated by a dielectric, such as interconnect or floating nets. When this capacitance is not part of a design, it is parasitic capacitance.

Parasitic capacitance couples the two conducting layers. This has the effect of slowing rise and fall times in a signal line, introducing noise, or possibly even causing charge leakage.

Some typical situations where parasitic capacitance may be a concern include the following:

- **Power line close to signal line**

Power lines carry large charge, and do not change frequently. Capacitance between a power line and time-critical interconnect can slow down a signal and cut in to timing

margins. Noisy signals can also couple noise into a power line, which then carries it to other nets.

- **Metal line over substrate**

In analog blocks, capacitance between substrate and antennas can cause excessive noise on the signal. This is particularly an issue in mixed-signal chips, where the substrate carries unwanted electrical signals from digital blocks.

- **Parallel interconnect**

When two signal lines run parallel, too much coupling can cause glitches, where a transition in one signal momentarily raises or lowers the voltage on the other line. This can pass bad data to the device on the other end.

- **Shielded signal**

A common technique when a clock is routed close to a sensitive signal line is to also route a line tied to ground or power in between. The tied line shields the sensitive signal line, but may also load the clock line, causing its transitions to be less sharp and skewing the clock cycles received in that section of the clock tree.

Capacitance Models in Parasitic Extraction

xCalibrate uses several models for calculations; xRC combines the xCalibrate models into coupled capacitance and intrinsic capacitance. In the netlist, these can be output separately or with coupled capacitance grounded, referred to as “lumped” capacitance.

xCalibrate Models

The xCalibrate rule file generator has both calibration models and effect models. In the xCalibrate transcript, references to “model 114” or “model 2024” refer to the calibration models. They are used with the technology description to create small layouts for the field solver. Results from the field solver are then analyzed by a curve fitter to determine coefficients for the capacitance and resistance equations.

The effect models are codified in the calibrated equations. As of 2008.1, the models are encrypted. They are occasionally referenced in xRC PDB transcripts. The capacitance models include the following:

- **Capacitance Connection** – models via and contact capacitance.
- **Capacitance Plate** – models capacitance between the lower surface of a wire and substrate (intrinsic), or the lower surface of a wire to the upper surface of another wire (crossover).
- **Capacitance Nearbody (NB)** – models capacitance between the sides of two wires, either on the same layer or different layers.

- **Capacitance Fringe** – models capacitance between the side of a wire and either the substrate (intrinsic) or the bottom or top of a wire (crossover).

These models are used for calculations. During formatting, the calculated values are consolidated into intrinsic and coupled capacitance. The intrinsic and coupled capacitance are output into the netlist and reports.

Intrinsic

Intrinsic capacitance is capacitance between an interconnect and substrate, sometimes referred to as ground. For lower layer interconnect, it is often more significant than capacitance between two lines.

Figure A-1. Intrinsic Capacitance

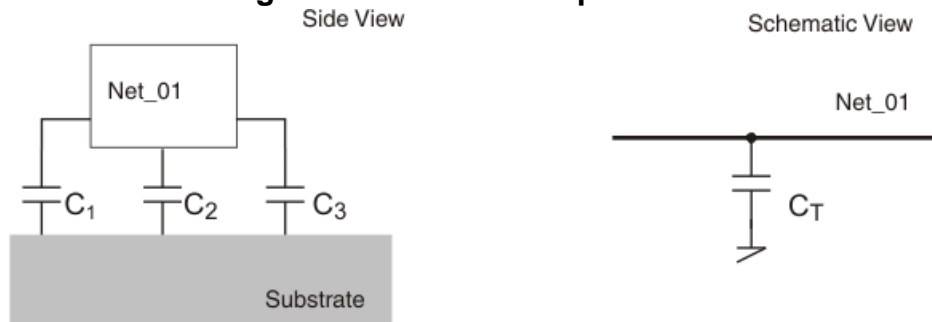


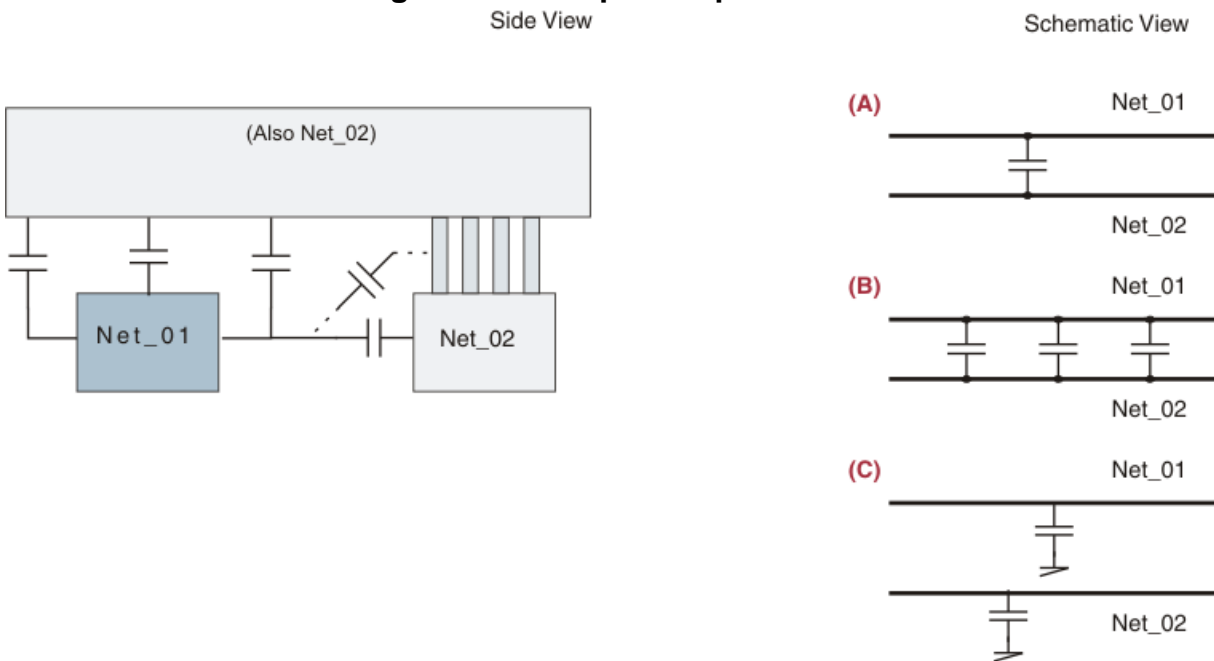
Figure A-1 shows a cross section and schematic view of intrinsic capacitance. The three parasitic capacitors shown in the layout represent the individual parasitic effects calculated by the calibrated rules. When the Calibre xRC formatter runs, the separate capacitances (C_1 , C_2 , and C_3 in the figure) are combined to a single intrinsic capacitance (C_T in the figure). Intrinsic capacitance appears in the netlist as “*ci_instance*”¹.

Coupled

Coupled capacitance is capacitance between two conductors. Both the conductors are usually interconnect. (Calibre xRC is not recommended for device extraction. Devices are typically represented with device models provided by your foundry.)

1. Before v2007.4, intrinsic capacitance was *c_instance*.

Figure A-2. Coupled Capacitance



As shown in [Figure A-2](#) coupled capacitance is calculated across layers. The contribution from vias is controlled separately. Most often, it is turned off because it contributes very little but causes extraction to take longer.

How the calculated capacitance is represented in the output depends on your Calibre xRC settings. [Figure A-2](#) (A) represents “capacitance only” output, which you would get from the -c extraction and formatter setting. The separate coupled capacitance effects are aggregated into a single net-to-net parasitic capacitance value. [Figure A-2](#) (B) represents “distributed” output, typical of -rcc extraction. The sum of the three coupled capacitors is the same as that of the single coupled capacitor in (A). The three capacitors are separated by parasitic resistors (not shown) along the net. [Figure A-2](#) (C) represents grounded, also sometimes called “lumped”, coupled capacitance. The total coupling capacitance is present on both nets and represented as going to ground.

Coupling capacitance is shown in a netlist as `cc_instance`.

Lumped

Lumped capacitance is a term loosely referring to aggregated parasitic capacitance. It is often applied to capacitance-only (-c) extraction because it outputs a single value for a net’s intrinsic capacitance and a single value for each net to which it is coupled.

Another situation sometimes referred to as lumped capacitance is when capacitance between nets is “broken” and “lumped to ground” as shown in [Figure A-2](#) (C). This can occur for several reasons:

- The Calibre xRC formatter is run with the “-g” switch.

- Extraction is run iteratively, and only one of the nets is in the more detailed extraction. (For example, an entire chip is extracted as “capacitance only,” followed by selected critical nets extracted with -rcc, and then the entire chip is netlisted.)
- A reduction such as PEX Reduce CC was applied. (Grounding coupled capacitance makes simulation faster because the two nets do not need to be kept in step.)

The total capacitance on a net is the same whether it is represented as separate intrinsic and net-to-net coupled capacitance, or as a single lumped value.

Parasitic Resistors

Parasitic resistance is an inherent property of any material. Conductors have very low resistance to current flow, but are not perfect. The unwanted resistance is calculated post-layout by parasitic extraction and then included in simulation because the parasitic resistance of interconnect can decrease signal amplitude and increase rise time. In power lines, parasitic resistance can also cause DC voltage drop.

Parasitic resistance runs along the conductive materials. Unlike capacitance and inductance, it is not influenced by nearby structures.

Some typical situations where parasitic resistance may be a concern include the following:

- **Non-symmetrical layouts for functionally equivalent objects**

When two instances of an object have unequal path lengths or widths, a signal reaches the ports at different times. The time difference can be enough to cause problems in differential pairs or intermittent faults.

- **Tight timing budgets**

Increases in signal rise time can cause signals to become skewed, eating into a timing budget. Parasitic resistance values make timing analysis simulations more accurate.

- **IR drop**

IR drop, also known as voltage drop, can seriously degrade a signal. It is common in high fanout and long, narrow paths for signals, as well as pervasive nets such as power and ground. You have to have enough power lines to supply your devices in a given area and enough drive strength to propagate the signal to all nets.

- **Electromigration (EM) analysis**

Electric current wears down conductors over time, known as electromigration. Parasitic resistance is an easy to calculate method of measuring how much the material resists - and will be worn down by - the current.

Resistance Models in Parasitic Extraction

The xCalibrate rule file generator produces rules for resistance calculations. The rules incorporate factors for multiple process corners and in-die variation. They provide the information needed by the Calibre xRC resistance engine, which recognizes shapes and applies different resistance calculations for each.

Some parts of the foundry-provided information can be overridden at run time on a per-layer basis. This is done through the [PEX Resistance Parameters](#) SVRF statement.


The resistance models include the standard parameters such as resistivity and temperature sensitivity (represented in netlists as either temperature coefficients or by modifying the reported temperature based on the SPICE calculation). They also include maximum length, used for breaking long shapes into smaller segments, and maximum area, used for breaking large areas into smaller ones.

Rho, Sheet, and Connection Resistance

The resistance of the conductive material is reported either as sheet resistance (ohms per square) or rho (bulk resistivity in ohm-meters). For vias and contacts an equivalent connection resistance is used. There are two reasons for this: current crowding caused by the small diameter of vias relative to the large wires they connect, and secondly because the different materials of the via and two metal layers disrupts electron flow.

Either rho or sheet resistance can be used with xCalibrate. They become equivalent resistance equations for metal layers and are encrypted. Calibrated rule files prior to version 2007.4 might include Resistance Sheet or Resistance Rho SVRF statements. Rule files that use Resistance Rho must also include a way to calculate layer thickness.

Note

 For processes below 90 nm, you should use resistance rules calibrated by xCalibrate v2008.1 or newer. These include better methods for handling layer thickness and bias than the older, manually created, rules.

Connection resistance applies wherever conductive layers overlap. It can be used to model connections created by traditional contacts and vias, or connections created by removing dielectric between process layers. Connection resistance, R , is based on the following equation:

$$R = \frac{n_1}{\text{common contact area of layer 1 and layer 2}} + \frac{n_2}{\text{common contact perimeter of layer 1 and layer 2}}$$

The SVRF file specifies n_1 and n_2 in the Resistance Connection statement. Typically, n_2 is 0.

Parasitic Resistor Representation

When you extract parasitic resistance (-r alone, or in combination with any of the other parasitic types) it is referred to as distributed extraction. This is because the interconnect is fractured into smaller sections and the parasitics are *distributed* along the wire. (Any other parasitics are connected at the nodes created by the fracturing.)

If your Calibre run includes reduction, the parasitic resistors may not appear to lie along the wires in a layout viewer. In cases of extreme reduction, parasitic resistors may even cross layers. They do not, however, cross devices or nets.

Parasitic resistance is shown in a netlist as `r_instance`.

MAXLENGTH and Fracturing

The resistance engine breaks grid-aligned rectangles into equal-sized smaller pieces to more accurately calculate resistance. Each piece maps to one parasitic resistor. (Plates and very wide wires whose width is greater than MAXLENGTH are split in both the x and y direction.) The largest possible segment is 100 microns; typically the segment size will be smaller.

MAXLENGTH also affects via clusters. If the area covered by a via cluster is longer or wider than MAXLENGTH, it will be divided into smaller groups. Groups of vias are modeled with representative parasitic resistors; the exact method depends on how the SVRF statement [PEX Reduce Via Resistance](#) is set. That statement controls the initial clustering and grouping of vias; those groups are then subject to the MAXLENGTH (or MAXAREA) value.

MAXAREA is intended for vias that cover large areas. Some foundries now offer solid vias between upper conductor layers where the single via covers an area equivalent to a large via array. The MAXAREA setting applies to this and other inter-layer connection types that rely on a large overlapped area rather than traditional vias.

Maximum segment size can be overridden with the SVRF statement [PEX Resistance Parameters](#) ... MAXLENGTH or in the Calibre Interactive interface with **PEX Options > Misc > Resistance Parameters**. The MAXAREA parameter does not affect plates and wires but only large via areas.

Appendix B

Parasitic Extraction Commands

Parasitic extraction has several specialized SVRF statements and controls.

This chapter describes some of the basic requirements of the SVRF rule file. It provides enough information for reading and creating a simple rule file.

The *Standard Verification Rule Format (SVRF) Manual* is the central document you consult for creating legal SVRF rules and operations for inclusion in your SVRF rule file.

The section “[About SVRF Rules](#)” provides information on how an SVRF rule file is interpreted. The remaining sections describe required and optional rules needed for primary extraction operations.

Note



You must use the same SVRF rule file for creating the PHDB, creating the PDB, and netlisting.

About SVRF Rules.....	173
Required Set.....	174
Required for Lumped Capacitance	177
Required for Distributed.....	178
Statements Specific To Distributed	179
Required for Source-Name Extraction	180
Example Pre-PEX SVRF File.....	181

About SVRF Rules

Guidelines to remember when working with SVRF rule files.

Order

SVRF rule files are compiled before processing. Except for nested variable declarations and conditional statements, you can write them in any order. For the example in this manual, the statements are organized according to function and the order in which the tool will process the statements.

Case Sensitivity

Keywords and names *are not* case-sensitive. You can set environment variables to make input case-sensitive, but by default it is not. Because of their relationship to the host platform's directory system, structure names and pathnames *are* case-sensitive.

Keywords Reserved

SVRF keywords are reserved words. Consequently, you must not use reserved words for names of variables or layers. For a complete list of reserved words, see the [Standard Verification Rule Format \(SVRF\) Manual](#).

Required Set

To create a PHDB, your rule file must have the following types of rules:

- Design specification: [Layer](#), [Layout Path](#), [Layout Primary](#), [Layout System](#), and [Mask SVDB Directory](#)
- Connectivity extraction: [Capacitance Order](#), [Connect](#), [Connect By](#), or [Stamp](#)

Note



If you are using coplanar layers, the [Capacitance Order](#) statement should be omitted from rule files generated by 2010.3 or later versions of xCalibrate.

- Intentional device recognition: [Device](#)
- [PEX Report](#) (even if you are doing only distributed netlists)

The rule summaries are listed in alphabetical order. Each links to the specific command in the [Standard Verification Rule Format \(SVRF\) Manual](#).


You can find in-depth usage information for the Calibre nmLVS-H tool SVRF rule requirements in the following Siemens EDA documentation:

- [Calibre Verification User's Manual](#)
 - [Connectivity Extraction](#)
 - [Device Recognition](#)
 - [LVS Circuit Comparison](#)
- [Standard Verification Rule Format \(SVRF\) Manual](#) for Calibre nmLVS-H specific SVRF rule file statements

Capacitance Order

The [Capacitance Order](#) statement defines the vertical order of the specified conductor layers from bottom-to-top—any capacitance layer must appear in the Capacitance Order statement. Therefore, you must specify any layer present in the capacitance statements in the Capacitance Order statement.

Note


 If you are using coplanar layers, the [Capacitance Order](#) statement should be omitted from rule files generated by 2010.3 or later versions of xCalibrate.

Capacitance Statements

The capacitance calculation statements are generated by the xCalibrate rule file generator and included in the main SVRF rule file. They are based on foundry-supplied process information and should not be modified by hand. The capacitance calculations specify what effects the Calibre xRC tool extracts.

For a discussion of parasitic elements and how they are modeled, see “[Parasitic Effects and Calibre Tools](#)”.

Note

 A design's nets must have resistance and intrinsic capacitance. Otherwise, the tool-produced net model could improperly represent the net.

Connect, Connect By, or Stamp

The [Connect](#), [Connect By](#), and [Stamp](#) statements direct connectivity extraction. Conductors must have connectivity.

Because Stamp does not provide the level of detail needed for extraction it should only be used for LVS. All layers necessary to connect parasitic layers to device layers should be involved in Connect (or SConnect) statements, which produce a valid point of connection for the device pins.

Because the Calibre xRC tool works from the design's electrical data, the PHDB creator performs connectivity extraction prior to parasitic extraction. Consequently, you must use at least one of these statements for specifying the connection between any abutting or overlapping objects.

The CONNECT BY statement enables the user to connect more than two layers, as follows:

```
CONNECT layer1 layer2 layer3 layer4 ... BY layerC
```

However, using this method to connect more than two layers impacts the performance of the resistance engine. Therefore, it is recommended that you do not use this method to specify the connection of more than two layers unless it is absolutely necessary.

Device

The [Device](#) statements define pins on low-level devices such as transistors and capacitors.

The Device statement directs device recognition. In the Calibre xRC tool, you use this statement for defining the device pins, which are necessary for extracting resistance values for distributed RC extraction.

Layer

The [Layer](#) statement defines the name of an original layer or an original layer set in terms of layer numbers or other original layer names in the rule file.

The amount of “hierarchy” within an original layer specification statement is unlimited. You may not, however, redefine original layers. Additionally, you cannot use the same name for multiple layer statements.

You must use Layer specification statements if you have original layer names in the rule file, and if you are referencing original layers by name in the Calibre xRC application.

Layout Path

The [Layout Path](#) statement specifies the pathname for your layout database. When you supply multiple Layout Path statements, the first database must contain the top cell you specify with the [Layout Primary](#).

Layout Primary

The [Layout Primary](#) statement identifies a top-level cell of your layout database for tool operation. When you specify a GDSII layout database, you must also specify the layout’s top-level cell using this statement.

Layout System

The [Layout System](#) statement specifies your layout database type. You can specify this statement once in the rule file.

Mask SVDB Directory

The [Mask SVDB Directory](#) statement specifies the directory location for creating the [SVDB](#). This database contains the [PHDB](#) and [PDB](#).

In your SVRF rule file, you must include this statement with the XRC secondary keyword using the following syntax:

```
MASK SVDB DIRECTORY filename XRC
```

where *filename* is a required string specifying an absolute or relative filename.

PEX Report

The [PEX Report](#) statement generates a report of parasitic results for the extracted circuit. Use this statement for any parasitic operation regardless of whether it is distributed RC, lumped C, or simple extraction.

Resistance Statements

You must have at least one SVRF Resistance statement in your rule file for specifying what values the Calibre xRC tool will extract. Your rule file may use the [Resistance Connection](#), [Resistance Rho](#), and [Resistance Sheet](#) statements, or include a foundry-supplied set of resistance statements in the file of calibrated rules.

Note



A design's nets must have resistance and intrinsic capacitance. Otherwise, the tool-produced net model could improperly represent the net.

For a discussion of parasitic elements and how they are modeled, see “[Parasitic Effects and Calibre Tools](#)”.

Required for Lumped Capacitance

When creating lumped capacitance netlists, your SVRF rules specify the following:

- The netlist type and format: [PEX Netlist](#) (required)
- How the tool performs reduction: [PEX Reduce Mincap](#)

The file must also include all the rules mentioned in “[Required Set](#)” on page 174.

Lumped Capacitance Netlist

The PEX Netlist statement controls several aspects of lumped C netlist creation including the following:


- The netlist's name.
- The netlist's format: DSPF, HSPICE, or Spectre. You can also specify a scaling factor.
- The name of the ground node.

[Example B-1](#) shows an example statement.

Example B-1. PEX Netlist Statement for Lumped Capacitance Extraction

```
PEX NETLIST "netlist.lumped" HSPICE 1e-6 SOURCENAMES GROUND VSS
```

Tip

 The “[PEX Netlist](#)” section of the *Standard Verification Rule Format (SVRF) Manual* explains usage of this statement in detail.

Specific To Lumped C

The reduction statement [PEX Reduce Mincap](#) is applicable to lumped capacitance. There is also a statement for creating reports, [PEX Report](#).

PEX Reduce Mincap

The Calibre xRC tool relies on [PEX Reduce Mincap](#) statement for controlling and reducing coupling effects. Use this statement to specify reduction and coupling thresholds for the tool.

Format the netlist using the “-c” Calibre xRC formatter invocation switch. For example:

```
calibre -xrc -fmt -c rule_file_name
```

This mode writes the entire contents of the PDB into a netlist and uses the PEX Reduce Mincap statement.


Lumped C Report

The [PEX Report](#) statement specifies the report’s name and the output format. [Example B-2](#) shows an example statement.

Example B-2. PEX Report Statement For Lumped C

```
PEX REPORT report.lumped SOURCENAMES
```

Tip

 The “[PEX Report](#)” section of the *Standard Verification Rule Format (SVRF) Manual* explains usage of this statement in detail.

Required for Distributed

Certain statements are required for distributed RC extraction.

When creating distributed RC netlists, your SVRF rules specify the following:

- The netlist type and format: [PEX Netlist](#) (required)
- Whether the tool performs reduction: [PEX Reduce TICER](#) and [PEX Reduce ROnly](#)
- Thresholds: [PEX Reduce Mincap](#)

The file must also include all the rules mentioned in [Required Set](#).

Distributed Netlist

The [PEX Netlist](#) statement controls several aspects of distributed RC netlist creation including the following:

- The netlist's name.
- The netlist's format: HSPICE, DSPF, SPEF, Eldo, or Spectre. You can also specify a scaling factor.
- Element reduction.


You specify the distributed RC netlist type and format using the [PEX Netlist](#) statement.

[Example B-3](#) shows an example statement.

Example B-3. PEX Netlist Statement for Distributed RC Extraction

```
PEX NETLIST "netlist.dist" HSPICE 1e-6 SOURCENAMES GROUND VSS
```

Tip

 The “[PEX Netlist](#)” section of the *Standard Verification Rule Format (SVRF) Manual* explains usage of this statement in detail.

Reporting Resistor Locations in the Netlist

You can report certain parasitic resistor-related information in the tool-produced netlist using the following optional keywords in conjunction with the PEX Netlist statement:

- RLOCATION — reports resistor locations.
- RWIDTH — reports resistor widths.
- RLAYER — reports resistor layers.

Statements Specific To Distributed

There are three additional commands: one to model in-die variation and two netlist reduction commands.

- [Parasitic Variation](#)
- [PEX Reduce TICER](#)
- [PEX Reduce ROnly](#)

In extracting distributed netlists, Calibre xRC also applies the [PEX Reduce Mincap](#) rule if it is in the rule file. Use [PEX Report](#) for creating ASCII reports.

Parasitic Variation

You can modify nominal resistance calculations with the [Parasitic Variation](#) statement to more accurately model your fabrication process and account for in-die variations.

PEX Reduce TICER

You can reduce the parasitic networks for a distributed RC or RCC netlist with the [PEX Reduce TICER](#) statement. This statement supports the Time Constant Equilibration Reduction method, known as TICER. This technique is described in detail in the section [TICER](#).

PEX Reduce ROnly

The [PEX Reduce ROnly](#) statement is the most aggressive reduction, but can be used only on pure resistive networks (that is, ones extracted with the -r switch, and not -rc, -rcc, or -c).

If this reduction and the TICER reduction are both specified, the Ronly reduction has precedence provided that the network is purely resistive. If the parasitics contain capacitance or inductance information, the TICER reduction is used.


Distributed RC Report

The [PEX Report](#) statement specifies the report's name and the output format. [Example B-4](#) shows an example statement.

Example B-4. PEX Report Statement for Distributed RC Extraction

```
PEX REPORT report.distrib SOURCENAMES
```

Tip

 The “[PEX Report](#)” section of the *Standard Verification Rule Format (SVRF) Manual* explains usage of this statement in detail.

Required for Source-Name Extraction

The following SVRF statements are required for source-name extraction:

- [LVS Report](#)

- [Source Path](#)
- [Source Primary](#)
- [Source System](#)

LVS Report

The [LVS Report](#) statement specifies the LVS report filename. You can specify this statement once in the rule file. Because this statement pertains to the Calibre nmLVS-H tool, you must include this statement when you use a source name extraction flow.

Source Path

The [Source Path](#) statement specifies the path to your schematic netlist database when performing source name extraction. You must specify this statement for subsequent use by the Calibre nmLVS tool.

Source Primary

The [Source Primary](#) statement specifies the design filename, subcircuit, or cell name for SPICE source databases when performing source name extraction. You must specify this statement for subsequent use by the Calibre nmLVS tool.

Source System

The [Source System](#) statement specifies your schematic netlist database format, specifically SPICE, when performing source name extraction.

Note



The Calibre xRC formatter will leave intact illegal syntax from your input source (for example, net names containing braces “{ }” in a SPICE input source). Consequently, you must ensure your input source is compatible with the legal syntax of the output netlist format.

Example Pre-PEX SVRF File

The following example SVRF rule file is a working example for connectivity extraction for Calibre nmLVS. It represents the minimum required functionality before adding Calibre xRC required statements.

```
////////////////////////////////////////
// DEFINE LAYOUT AND SOURCE INPUT
//      (this can also be done in Calibre Interactive)
////////////////////////////////////////

LAYOUT PATH "two_xtors.gds"
LAYOUT PRIMARY "top"
LAYOUT SYSTEM GDSII

SOURCE PATH "source.sp"
SOURCE PRIMARY "top"
SOURCE SYSTEM spice

////////////////////////////////////////
// IDENTIFY BASE LAYERS FOR FASTER DEVICE RECOGNITION
////////////////////////////////////////

LAYOUT BASE LAYER contact pplus nplus poly oxide pwell

////////////////////////////////////////
// MAP DRAWN LAYERS
////////////////////////////////////////

LAYER pwell           1
LAYER oxide           2
LAYER poly            4
LAYER nplus           5
LAYER pplus           6
LAYER contact         7
LAYER metall          8
LAYER via             9
LAYER metal2         10

////////////////////////////////////////
// ATTACH TEXT LAYERS
////////////////////////////////////////

TEXT LAYER 50
ATTACH 50 metall
ATTACH 50 metal2
LABEL ORDER metall metal2
```

```

////////////////////////////////////
// DERIVE BOOLEAN LAYERS
////////////////////////////////////

bulk    = EXTENT
substr  = SIZE bulk BY 2
nsub    = substr NOT pwell

narea   = oxide AND nsub
pgate   = narea AND poly
pox     = oxide AND pplus
psd     = pox NOT poly
parea   = oxide AND pwell
ngate   = parea AND poly
nox     = oxide AND nplus
nsd     = nox NOT poly

////////////////////////////////////
// CONNECT OPERATIONS
////////////////////////////////////

SCONNECT psd pwell
SCONNECT nsd nsub
CONNECT metall poly nsd psd BY contact
CONNECT metall metal2 BY via

////////////////////////////////////
// DEFINE INTENTIONAL DEVICES
////////////////////////////////////

DEVICE mn ngate poly nsd nsd pwell [0]
DEVICE mp pgate poly psd psd nsub [0]

////////////////////////////////////
// SHORT ISOLATION (optional but recommended)
////////////////////////////////////

LVS ISOLATE SHORTS YES BY LAYER ALSO BY CELL ALSO

//LVS EXECUTE ERC YES
//ERC SELECT CHECK "check names here"
//ERC RESULTS DATABASE "erc.db"

//LVS SOFTCHK parea LOWER

////////////////////////////////////
// INDICATE POWER AND GROUND NET NAMES
////////////////////////////////////

LVS POWER NAME "vdd?"
LVS GROUND NAME "vss?"

```

```
////////////////////////////////////  
// SPECIFY OUTPUTS (SVDB AND LVS REPORT)  
//      (this can also be done in Calibre Interactive)  
////////////////////////////////////  
  
MASK SVDB DIRECTORY "svdb" QUERY //XRC  
LVS REPORT "lvs.rep"  
  
////////////////////////////////////  
// LVS COMPARISON SECTION  
////////////////////////////////////  
  
LVS RECOGNIZE GATES ALL  
LVS REPORT MAXIMUM 50 // ALL
```


Appendix C

Output Reference

Calibre xRC generates various forms of extraction data and output.

This appendix provides reference information on the various types of data which the Calibre xRC tool produces.

Databases	186
Logfiles	189
Netlists.....	190
Reports	193
Templates	195

Databases

Calibre xRC uses databases to store design and extraction data at different stages of the extraction process.

There are three databases:

- [Parasitic Database \(PDB\)](#)
- [Persistent Hierarchical Database \(PHDB\)](#)
- [SVDB](#)

Parasitic Database (PDB)

Each PDB contains the results of parasitic extraction and analysis on a specified primary layout cell and the data for the design's total nets. The Calibre xRC formatter generates netlists and reports from the data a PDB contains.

A PDB can only be created by running the Calibre xRC invocation with the -pdb switch.

PDB Contents

The PDB stores the parasitic models for each extracted net. These models consist of the following elements:

- The net's name
- The collection of the device pins, ports, parasitic delays, and circuit elements.

A PDB also includes port, device pin, instance pin, and circuit element locations.

PDB Segment File

Each PDB contains a segment file named pdb.seg. This file consists of a list of segments containing the parasitic models. By default, each PDB contains four segments, each holding up to 2.147 gigabytes of data.

Example C-1. PDB Segment File

```
// This segment file was automatically created by libpdb.a
// Use the file pdb.dat for data, and use max size for segment
pdb1.dat
pdb2.dat
pdb3.dat
pdb4.dat
```

If you know the tool will produce extraction data greater than 8.588 gigabytes in size, you can create the *pdb.seg* file before executing PDB generation. Using a text editor, perform the following steps:

1. In the *pdb.seg* file, list the four files in the PDB database directory and the additional files you want, beginning with *pdb5.dat*.
2. Save the file to a directory named *layout_primary.pdb* or *cell_name.pdb* in your SVDB directory.

layout_primary.pdb is the name specified by the [Layout Primary](#) specification statement in the rule file.

Persistent Hierarchical Database (PHDB)

A PHDB can be created with either Calibre nmLVS or Calibre xRC software. Although you can use either LVS or LVS-H (hierarchical LVS), all parasitic extraction except transistor level requires a hierarchical database; that is, you cannot use non-hierarchical Calibre nmLVS except for transistor-level extraction.

The Calibre xRC tool requires and uses the PHDB for the second stage of the parasitic extraction process: Parasitic Database (PDB) generation. The tool stores the PHDB in the [SVDB](#).

Generally, to produce output that uses source names, you create the PHDB using the Calibre nmLVS-H tool. To produce output that uses layout names, you create the PHDB using the Calibre xRC tool.

PHDB Contents

The PHDB stores selected hierarchical geometries and the results of connectivity extraction and device recognition. A PHDB contains the following information:

- Hierarchical connectivity model: net placements
- Hierarchical geometry model: primitive device placements

The recognized devices determine current flow and their placement determines where nets end.

PHDB Compared To Calibre nmLVS-H Hierarchical Database

The PHDB is equivalent to the Calibre nmLVS-H Hierarchical Database (HDB). The HDB is an in-memory database. Consequently, you regenerate the HDB with each Calibre nmLVS-H run.

In contrast to the Calibre nmLVS-H HDB, you only need to regenerate the PHDB for the following reasons:

- You change your layout.
- You change your xcell file.
- You change your SVRF rule file.

On regeneration, the Calibre xRC or Calibre nmLVS-H tool replaces the existing PHDB with the new PHDB.

SVDB

The Standard Verification Database (SVDB) is a central repository for data files and directories the Calibre nmLVS-H and Calibre xRC tools create, specifically the PHDB and PDB.

Additionally, the SVDB stores the cross-reference files the Calibre nmLVS-H tool creates and the Calibre xRC formatter uses when you use source name extraction.

SVDB Contents

During PHDB and PDB creation, the Calibre nmLVS-H and Calibre xRC tools generate the SVDB directory's contents in separate databases. By default, the SVDB stores the following files:

- **Layout netlist:** The Calibre nmLVS-H tool or command generates the layout netlist. The layout netlist contains connectivity data for the top-level cell and subcircuits down to the primitive device level. Additionally, the layout netlist describes connections for ideal nets, specifically nets without parasitic elements (ideal conductors).
- **Cross-reference files:** Required files when you perform source name extraction. The Calibre nmLVS-H application generates cross-reference files. You must run the Calibre nmLVS-H application *before* the Calibre xRC tool.

The cross-reference files establish correspondence between the layout and source. The [Layout Primary](#) statement supplies the prefix for the files, and these files have the following extensions: *.extf*, *.ixf*, *.lph*, *.lvsf*, *.nxf*, and *.sph*. For a description of each file see the [Mask SVDB Directory](#) statement in the *Standard Verification Rule Format (SVRF) Manual*.

- [Persistent Hierarchical Database \(PHDB\)](#) directory. This directory is named *layout_primary.pdb*.
- [Parasitic Database \(PDB\)](#) directory

Specifying the SVDB

You specify the SVDB directory's location in your SVRF rule file using the Mask SVDB statement.

Logfiles

Logfiles generated during extraction contain valuable processing and results information.

Formatter Log PDB Net Summary

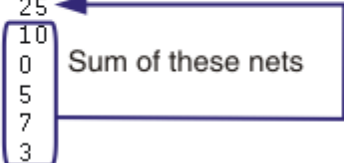
Example C-2 provides an example PDB Net Summary. The Calibre xRC formatter produces this summary in the formatter log, normally *fmt.log*, after completing a formatter run.

Example C-2. Formatter Log PDB Net Summary

```

-----
                                PDB NET SUMMARY
-----
pdb file name =                ../svdb_dir/TRAINING.pdb
root cell name =               TRAINING
total nets =                   25
top-level nets =               10
non-top-level nets =          0
degenerate nets =              5
merged nets =                  7
error nets =                   3

```



The diagram illustrates that the total number of nets (25) is the sum of the individual categories: top-level nets (10), non-top-level nets (0), degenerate nets (5), merged nets (7), and error nets (3).

Most of the fields in the PDB Net Summary are self explanatory; this section highlights the following cryptic parts of the PDB Net Summary:

- degenerate nets

A degenerate net is a net whose parasitic model contains no resistors or capacitors, just pins and ports. The Calibre xRC tool reports degenerate nets in the formatter transcript. Usually, the Calibre xRC tool identifies degenerate nets found in common source drain regions ([Figure C-1](#)) and connections by abutment ([Figure C-2](#)).

- merged nets

The Calibre nmLVS tool may cross-reference multiple layout nets to a single source net. In this case, the tool merges layout nets and writes the combined net model to the output netlist.

- error nets

The error nets' total represents the overall number of nets the Calibre xRC tool extracted and, consequently, identified with an error message.

Figure C-1. Common Source/Drain Region

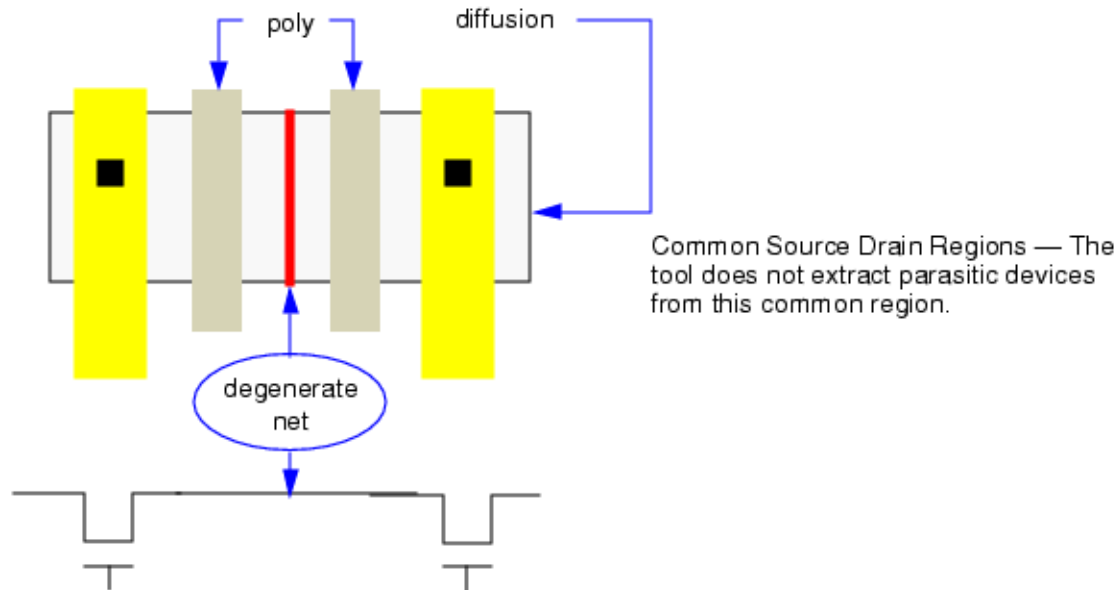
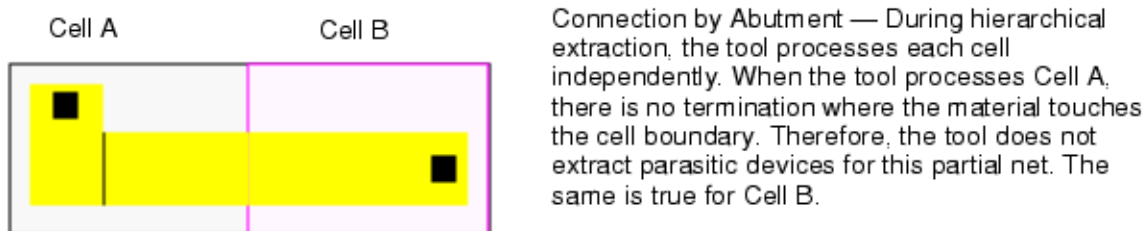


Figure C-2. Connection by Abutment



Netlists

The Calibre xRC tool outputs different files depending on the netlist format (for example, HSPICE) you specified. Normally, these files are in your current working directory.

- [HSPICE and Spectre Output Files](#)
- [DSPF and SPEF Output Files](#)
- [R-Coupled C DSPF Netlist Extraction](#)
- [Interpretation of _noxref Entries](#)
- [Intentional Device Property Handling for Seed Promoted Devices](#)

HSPICE and Spectre Output Files

The Calibre xRC formatter produces the following files during HSPICE or Spectre netlist creation, where *name* is the name specified in the PEX Netlist statement:

- *name* — the top-level netlist. If extracted hierarchically, this file contains the top-level.subckt definition, including intentional devices and *cell* instantiations. The *name.pex* file will be included outside the top cell subckt definition. The *name.top-level_cell_name.pxi* files will be included inside the top cell subckt definition. Flat extraction will not contain cell instantiations.
- *name.pex* — the parasitic model, intrinsic capacitor, and resistor netlist. This file contains the parasitic model subcircuit definitions for each extracted net. The .subckt name is based on the net's hierarchical connectivity. It also contains parasitic resistors and intrinsic capacitors. The name of each parasitic model is in the form *cell_name%net_name*.
- *name.top-level_cell_name.pxi* — the parasitic model instantiation and coupling capacitor netlist. This file contains instances of the parasitic models and describes the net and any pins or ports connected to the net, as well as any extracted coupling capacitors. The *top-level_cell_name* comes from the [Source Primary](#) statement's parameter defined in the rule file. If extracted hierarchically with the -full option, an *xcell_name.pxi* file is generated for each of the xcells in the design.

For lumped capacitance (-c) extraction, the *name.pex* file is not created and the parasitic intrinsic capacitors are written to the top-level netlist.

The Spectre-format netlist allows mixed-case property names. In order to preserve the case you must set the [Layout Case](#) or the [Source Case](#) statements in the rule file to YES. There are limitations in the LVS SPICE reader. For example, for built-in devices, the case for default property names are not preserved. For more information on SPICE see "[SPICE Format](#)" in the [Calibre Verification User's Manual](#).

A Spectre-format netlist generated by the Calibre xRC tool does not run with the Cadence Spectre simulator version 6 or newer because of case sensitivity. Include the following statement in the extracted netlist to resolve this:

```
simulator lang = spectre insensitive = yes
```

DSPF and SPEF Output Files

For DSPF and SPEF, the Calibre xRC formatter outputs a single netlist called *name*, where *name* is the value you specified for the output netlist file in the Calibre Interactive PEX interface or by the applicable PEX Netlist specification statement in the SVRF rule file. This netlist contains the parasitic models.

R-Coupled C DSPF Netlist Extraction

The Calibre xRC formatter outputs the coupling capacitors for an R-coupled C (RCC) DSPF netlist in a separate include file with the suffix “.cci”; for example, *top_cell.SIMPLE.cci*.

Interpretation of `_noxref` Entries

Under certain circumstances, the Calibre xRC tool appends `_noxref` to layout names in the netlist.

The Calibre xRC formatter appends `_noxref` to layout names when the layout name does not have an entry in the cross-reference file produced by the Calibre nmLVS-H run. This does not mean that the Calibre nmLVS-H run was incomplete or incorrect.

In general, the Calibre xRC formatter outputs these appended layout names for either of the following reasons:

- You enable gate recognition in the Calibre nmLVS-H tool. Normally, the `_noxref` names are internal to the gate.
- You enable parallel gate reduction in the Calibre nmLVS-H tool (for example, using the [LVS Reduce](#) statement).

The Calibre xRC formatter appends `_noxref` to these net names and propagates the annotated names into the netlist for the following reasons:

- The net represents a valid connection.
- The layout net name, if it were not annotated, could overwrite a legitimate source name with the same name. For example, if `net_01` is a source name and `net_01` is a layout name without an LVS cross-reference, the Calibre xRC formatter could short the two nets.

Use the [PEX Netlist Noxref Net Names](#) statement to eliminate `_noxref` net models from the netlist.

Intentional Device Property Handling for Seed Promoted Devices

To address seed promotion in your design use the [LVS Push Devices Separate Properties](#) YES statement together with the [LVS Push Devices](#) YES statement. These statements force seed promoted devices back down in most cases and write instance specific property values to a system determined file in the SVDB directory when necessary. The Calibre xRC formatter reads this system file and uses these property values to produce the extracted netlist.

For hierarchical netlists (-full), the Calibre xRC formatter produces a parasitic extraction output netlist in which each intentional device gets the average of the property values for that device across all xcell instances.

If no single property value for a specific pushed down device is correct for that device across all instances of an xcell, then an xcell instance-averaged property value is used to push down into the xcell for that specific device instead of randomly choosing a device property value from one particular instance of the xcell.

For gate-level extracted netlists, you can control the netlisting of the intentional circuit content of xcells in one of the following ways:

- On an all xcell basis using the [PEX Ideal Xcell](#) statement.
- On an individual xcell basis either by using the -I xcell flag in an xcell file or by using the IDEAL keyword in the [PEX Xcell](#) statement.

In both cases, the intentional circuit contents of xcells are netlisted without parasitics, treating the nets as ideal conductors. The xcell instance-averaged device property values are applied to the devices within the xcells whose intentional circuit contents are netlisted.

For flat netlists, the instance specific device property values are applied directly to the appropriate device instances.

Reports

The Calibre xRC formatter normally creates the ASCII report in your current working directory.

You control the report's name using either the [PEX Report](#) or [PEX Report Netsummary](#) statements in your rule file. The Calibre Interactive PEX interface overrides these with the **PEX Report File** field, under the **Reports** tab of the **Outputs** screen.

Distributed

The distributed report is produced only when a distributed netlist is specified in the formatter stage. The formatter transcript indicates a distributed report is output with the following line:

```
--- WRITING ASCII REPORT
```

The distributed report provides details on a net-by-net basis for parasitics. It reports the same type of information as the lumped report.

Lumped

The lumped report is produced only when a lumped capacitance (-c) netlist is specified in the formatter stage. It does not support source names. The report includes the following information:

- **Netid** — The numeric identifier used by the software.
- **R(UpperBound)** — An upper bound on resistance. Because this report is only produced for capacitance, the value is always 0.0.

- **Cvalue** — The total capacitance for the listed net. The units are farads.
- **%Coupled** — How much of the total capacitance is coupled capacitance versus capacitance to ground. Reduction occurs *before* this value is calculated making it a lower limit on the amount of coupling on the net.
- **RC(UpperBound)** — The value is always 0.0.
- **Netname** — The layout name of the net if it exists.
- **Net Details** — For each net, details of the following:
 - List of coupled nets by layout name.
 - Intrinsic capacitance in farads. This includes any decoupled capacitance which has been lumped to ground.
 - Coupled capacitance values by net in farads. If the PDB includes distributed parasitics, nets may appear multiple times.

Net Summary

The net summary report is produced when the rule file contains a PEX Report Netsummary statement. It contains data on the parasitic capacitance of the nets that were extracted. Unlike the lumped report, it is configurable. Reported values are multiplied by the SCALE option in the SVRF statement. If SCALE is not specified, the values are in farads.

The default format reports the following post-reduction information for all extracted nets:

- **GID** — An internal identifier for the net which remains the same as the net moves through different levels of design hierarchy.
- **totalC** — Total capacitance.
- **totalCC** — Total coupled capacitance. Because this value is post-reduction it represents a lower bound on the coupling of the net. For RC extraction this will always be 0 because all coupled capacitance is decoupled.
- **ratioCC** — The ratio of coupled capacitance to total capacitance. Reduction occurs before this value is calculated, making it a lower limit on the amount of coupling on the net.
- **Cell** — The cell the net is located in. Reflects cells in the xcell list only.
- **Layout** — The layout name of the net.
- **Source** — The source name of the list. When the PHDB is created using the -xrc -phdb switches instead of LVS, the report shows “<noref>” because the source names are not available.

If the report command has COLUMNS ADVANCED set, capacitance values include subtotals for top of cell versus subcells. The values are reported in the following columns:

- **localC** — The total capacitance of the portion of the net within the top of the named cell.
- **localCC** — The total coupled capacitance of the portion of the net within the top of the named cell.
- **childC** — The total capacitance of the portion of the net within subcells. For transistor-level and gate-level extraction this will always be 0.
- **childCC** — The total coupled capacitance of the portion of the net within subcells. For transistor-level and gate-level extraction this will always be 0.

Resistance Point-to-Point

The resistance point-to-point report is produced when the rule file contains a [PEX Report Point2Point](#) statement. It contains point-to-point parasitic resistance calculations of specified nets. These resistance calculation commands are specified in an input file.

The report is generated from a PDB. You can extract a report from a flat or hierarchically generated PDB. You can generate a point-to-point resistance report for a net at the top level cell or a net inside a cell represented in the xcell list. You can also generate a report for point-to-point resistance on a net based on coordinates in the layout. You cannot generate a report for a net that traverses the hierarchy of the PDB.

Net-to-Net Coupling Capacitance

The net-to-net coupling capacitance report is generated when the rule file contains a [PEX Report Coupling Capacitance](#) statement. The report contains a list of calculated coupling capacitances between nets in the layout.

The report shows the ratio of coupling capacitance between any two nets related to the total net capacitance for each net. You can generate the report based on a hierarchical or transistor-level extraction.

Templates

Templates are ASCII text files used for defining pin direction and order in the output netlist. They are created by the formatter in the SVDB directory, and can be edited in a text editor. (Older versions of the Calibre xRC tool wrote templates in the current working directory and the formatter checks that location first.)

Templates are enabled by setting the PEX_FMT_HP_PORT_MAP_MODE environment variable to one of the following mutually exclusive values:

- **TEMPLATE** — Maps ports using the user-defined template for each cell.

- **TEMPLATE CELLNAME SOURCE** — Defines an alternate name for the cell in the netlist. The Calibre xRC formatter writes cell names to the output netlist reflecting the contents of Template Cell Name (see [Figure C-3](#)). If the netlist naming mode is SOURCE in the PEX Netlist statement, then the formatter cross-references the Layout Netlist Cell Name to the schematic netlist (source netlist), and the tool uses the source cell name for the Template Cell Name.

Do not use the templates to manipulate port names. If you do, the formatter issues a warning message similar to the following:

```
WARNING:unable to find a map for port ORIG_NAME on the template for cell  
CELL ORIG_NAME will NOT be ported from the cell.
```

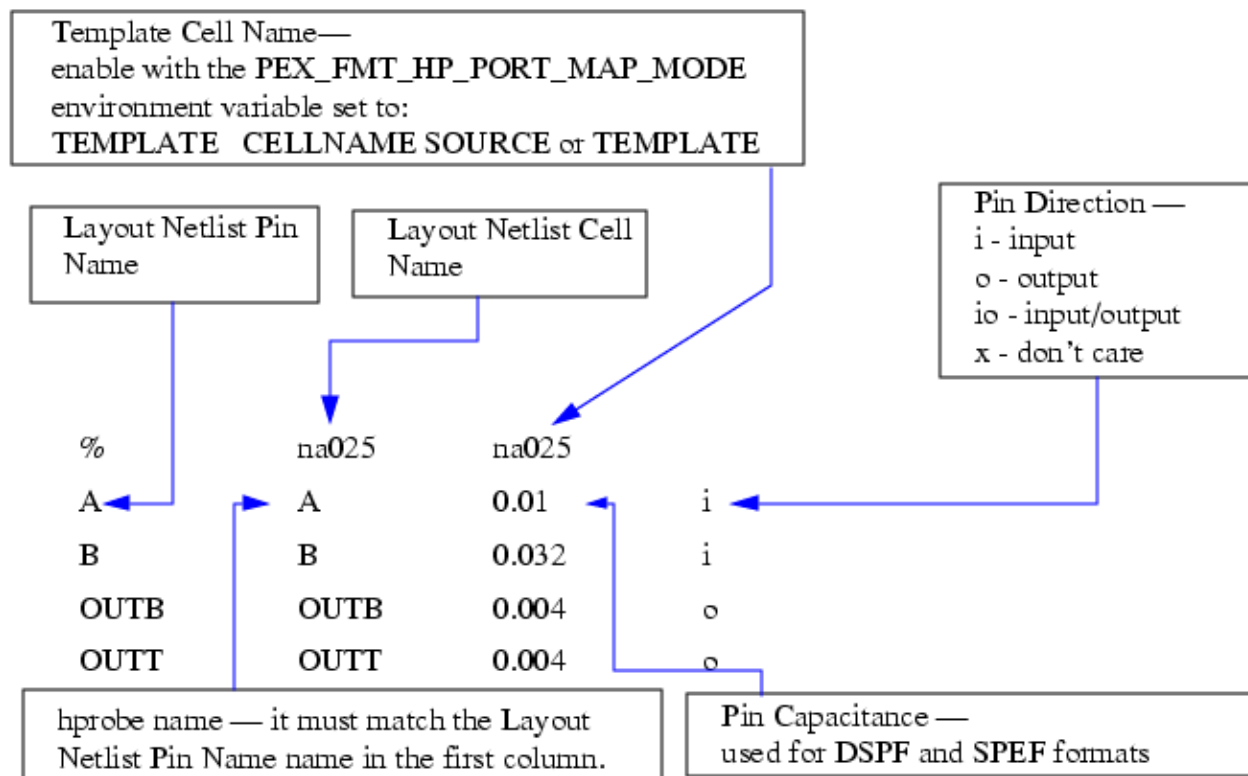
If you see this warning, the netlist is incorrect. To change port names, use the [Layout Rename Text](#) statement.

Once you enable templates and run the Calibre xRC formatter, the application searches first the SVDB directory and then the working directory for previously-defined template files. For any cell without a template file, the Calibre xRC formatter generates a default template and writes the template to the template directory (normally *./template*) in the SVDB directory. The formatter derives the default template's contents from a cell's interface description contained in the Layout Netlist.

The Calibre xRC formatter generates one template for each xcell and stores them in the templates directory. The resulting template files are named *xcell_name.stl*.

Figure C-3 illustrates the default components of an example Calibre xRC tool-generated template:

Figure C-3. General Template Structure



Appendix D

Reduction Techniques

Calibre xRC provides reduction controls used to reduce run time and/or netlist size.

This appendix provides information on the various types of reduction techniques available for parasitic extraction.

Capacitive and Resistive Reduction	199
Threshold-based Reduction	200
TICER.....	201

Capacitive and Resistive Reduction

Use reduction to control the number of capacitors and resistors that are output by extraction.

There are SVRF statements that reduce the number of capacitors and SVRF statements that reduce the number of resistors:

- **PEX Reduce CC** — Reduces the number of coupled capacitors by converting coupled capacitors that meet some constraint to lumped capacitance. The lumped capacitance on a net is represented as a single value coupled to ground, thus reducing the overall netlist size.
- **PEX Reduce Mincap** — Reduces both intrinsic and coupled capacitors based on a user-defined threshold value. The command can specify to remove or combine capacitors.
- **PEX Reduce Minres** — Reduces parasitic resistors by combining them based on a user-defined threshold value. The command combines parasitic resistors.
- **PEX Reduce Parallel** — Reduces two or more resistors connected in parallel between the same pair of nodes into a single equivalent resistor in the extracted netlist.

Note



The PEX Reduce CC reduction overall provides the best control. It is the only one which bounds the error that can be introduced by aggressive reduction.

PEX Reduce CC

The PEX Reduce CC specification is applied during netlisting. The total coupling capacitance between two nets is compared to a constraint, either an absolute value such as 3 femtofarads, or a percentage of the total net capacitance for either net. If the coupling capacitance is less than

the constraint, it is decoupled from the nets and included in the lumped capacitance to ground. Note, the percentage constraint must hold for both nets.

PEX Reduce CC runs before PEX Reduce TICER. For more information, see “[PEX Reduce CC](#)” in the *Standard Verification Rule Format (SVRF) Manual*.

PEX Reduce Mincap

The PEX Reduce Mincap specification is applied after the PDB generation, during the formatting stage. PEX Reduce Mincap implements two types of reduction, merging and removing, based on a user defined threshold.

By setting the REMOVE threshold, any capacitors that fall below the threshold value, are either grounded or removed.

By setting the COMBINE threshold, any capacitors that fall below the threshold value are combined with neighboring capacitors on the same net.

For more information, see “[PEX Reduce Mincap](#)” in the *Standard Verification Rule Format (SVRF) Manual*.

PEX Reduce Minres

The PEX Reduce Minres specification is applied after the PDB generation, during the formatting stage. PEX Reduce Minres COMBINE merges resistors based on a user-defined threshold.

For more information, see “[PEX Reduce Minres](#)” in the *Standard Verification Rule Format (SVRF) Manual*.

PEX Reduce Parallel

The PEX Reduce Parallel specification controls whether or not resistors in parallel are combined in the netlist during the formatting stage.

For more information, see “[PEX Reduce Parallel](#)” in the *Standard Verification Rule Format (SVRF) Manual*.

Threshold-based Reduction

Threshold-based reduction allows you to control the parasitic reduction output.

The [PEX Reduce Digital](#) statement decreases the size of netlists and databases. When you use this statement in your rule file, the Calibre xRC tool uses the threshold you define for distributed RC parasitic extraction; if a distributed RC model meets the threshold, then the Calibre xRC formatter will convert the model into a lumped C model by discarding the resistors.

TICER

TICER stands for “Time Constant Equilibration Reduction”. You specify this reduction method in your SVRF rule file using the following SVRF statement and keyword:

```
PEX REDUCE TICER frequency
```

where *frequency* is a user-defined calculated number controlling which nodes in the circuit the tool can select for subsequent elimination; specifically, the tool will select nodes with time constants less than the frequency parameter.

In your rule file, you specify the calculated frequency parameter in hertz and express the value in the SVRF PEX Reduce TICER statement using scientific notation. For example:

```
PEX REDUCE TICER 40e9
```

Using the Calibre Interactive PEX interface, you specify TICER reduction by selecting “Enable TICER reduction below” and entering a frequency. The “Enable TICER reduction below” option is in the PEX Options pane. To enable PEX Options, select **Setup > PEX Options** in the Calibre Interactive - PEX menu.

How TICER Works

The TICER reduction method preserves the frequency response of an RC interconnect circuit from dc up to the *frequency* parameter. Consequently, the *frequency* parameter provides you with a control for trading off compression with accuracy.

- Setting the *frequency* parameter to a higher value results in larger (more R and C elements) interconnect circuits having a wider bandwidth of accuracy.
- Setting the *frequency* parameter to a lower value results in more compression and an earlier roll-off in accuracy.

Calculating the *frequency* Parameter

You must calculate the frequency parameter using the following formula:

$$frequency = \frac{tradeoff_value}{transition_time_minimum}$$

where:

tradeoff_value — a number between 4 and 10. A larger tradeoff_value results in a higher frequency parameter value and, consequently, more accurate reduction results.

transition_time_minimum — the shortest rise or fall time you expect in your design. In general, you can estimate this value using 1/5 of your design’s switching delay.

TICER and Temperature Sensitivity Effects

If you include temperature sensitivity in the extraction process by using the PEX Temperature SVRF statement, the reduced netlist will include parasitic capacitors that have small changes in value due to temperature coefficients.

These changes are caused by how RC delays are calculated when using TICER. Including temperature coefficients in the extraction process affects the final values of parasitic resistors. During TICER reduction the values for resistors and capacitors are recalculated so that the RC delay in the network remain unchanged. Including temperature coefficients for resistors affects the recalculated values for the parasitic capacitors in the final netlist.

Appendix E

Time-it Tool Overview

The Time-it™ tool is a path-trace delay calculator, and computes both interconnect and cell delay data for a netlist.

This chapter contains information about the Time-it delay calculator.

Time-it Documentation	203
Time-it Application Overview	203

Time-it Documentation

The Time-it tool is a path-trace delay calculator, and computes both interconnect and cell delay data for a netlist.

This chapter presents a brief overview of the Time-it tool. For more information on using the tool, refer to the Time-it manual set, which is included in the Time-it software tree. The Time-it manual set includes the following documentation:

- *Time-it Reference Manual*
- *Time-it Release Notes*
- *Time-it Release Highlights*
- *Time-it Application Notes*

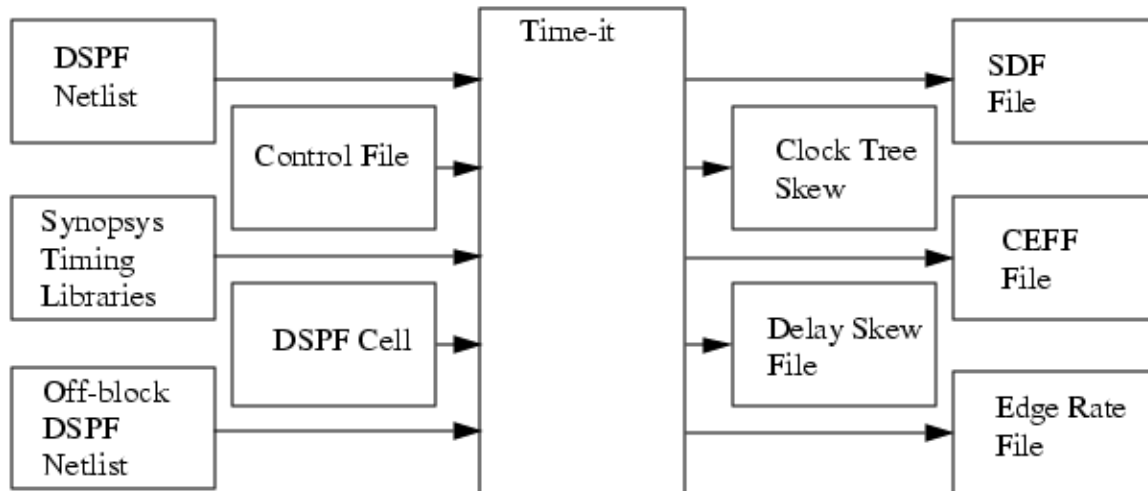
Time-it Application Overview

The Time-it tool is a high-capacity, path-trace delay calculator for deep submicron process cell-based designs.

Using the Time-it tool, you can perform accurate signal delay calculations using both cell and interconnect delay information at orders of magnitude faster than traditional simulators. This tool calculates the input pin-to-output pin delay for cells, and the output cell-to-input cell delay for interconnects.

[Figure E-1](#) illustrates the Time-it tool's inputs and outputs.

Figure E-1. Time-it Tool Design Flow



Time-it Tool Inputs

For inputs, the Time-it tool takes a cell-level and an RC interconnect netlist (or netlists), a Synopsys cell timing library, and a control file. You specify Time-it technology-dependent parameters in the control file, which consists of a set of command lines.

The cell-level netlist can be DSPF. The interconnect netlist should be in DSPF format; if you do not specify DSPF, the application uses wire load models specified in the timing library. Traditionally, the cell-level and interconnect netlists are in one DSPF file.

The timing library must be in Synopsys's ".lib" format.

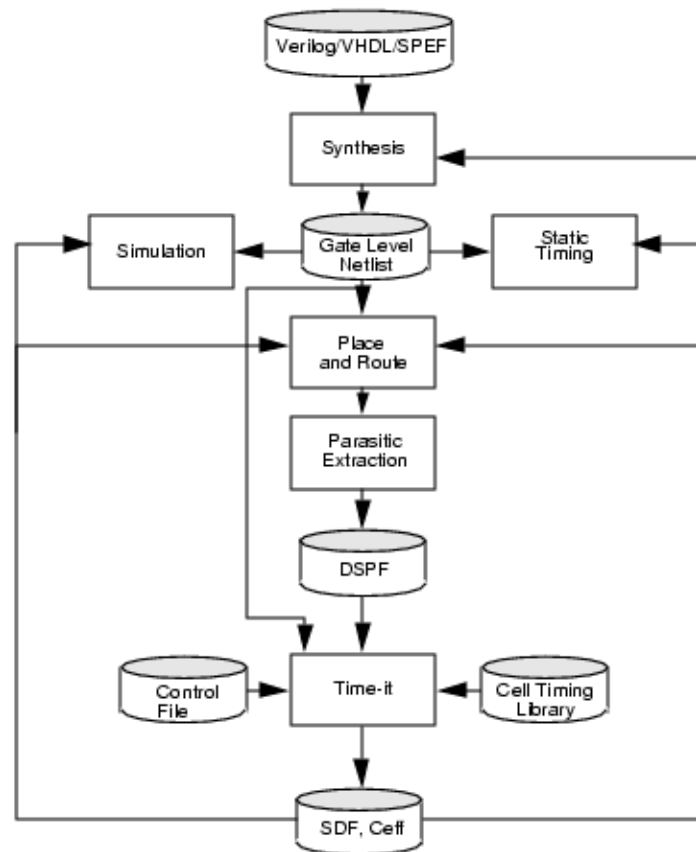
Time-it Tool Outputs

The Time-it tool outputs a Standard Delay Format (SDF) file. This file includes any cell instance pin-to-pin delays and interconnect driver-to-receiver delay, and timing check values and path constraints. A synthesis or timing analysis tool subsequently uses this netlist.

Additional outputs from the Time-it tool include a list of effective capacitances for each net (CEFF), the worst case rise and fall times of each net's drivers and receivers, a worst case skew report for each net, and a full clock tree skew report. You specify input and output parameters in the control file.

How the Time-it Tool Fits Into the Design Flow

Figure E-2 illustrates using the Time-it tool's outputs for quickly identifying potential timing problems in a design. The unintentional parasitic effects of interconnect cause these problems, for example slow nodes or excessive skew.

Figure E-2. Time-it Tool in the Design Flow

Simulators or static timing analyzers can use the SDF output for backannotation and perform further detailed timing verification or optimization. SDF is a standard format read by many Verilog or VHDL logic simulators, static timing analyzers, synthesis, and place and route tools.

Importance of Accurate Delay Calculation

As deep submicron design integration density and operating frequency increase, the effect of interconnect delays on overall system performance also increases. Specifically, interconnect delay effects on a design are important to performance. In addition, cell and interconnect delays are interdependent.

For accurately calculating the delay for a cell driving a distributed RC network, the delay tool must account for the interaction between the interconnect and the cell. In addition, for the most accurate delay calculations, the delay tool must calculate the effective capacitance (rather than the total capacitance) of the interconnect.

Appendix F

Error and Warning Messages

Error and warning messages are generated by the Calibre xRC tool at run time.

Error Messages 207

Warning Messages 208

Error Messages

Error messages must be corrected to continue a Calibre xRC run.

Table F-1. Calibre xRC Error Messages

Message	Possible Causes
A LAYOUT PRIMARY is not specified in the rule file.	Layout Primary statement is not specified in the rule file.
A LAYOUT PATH is not specified in the rule file.	Layout Path statement is not specified in the rule file.
Cannot determine the LAYOUT SYSTEM from rule file <i>filename.rule</i>	Layout System statement is not specified in the rule file.
MASK SVDB DIRECTORY was not specified in the rules file.	Mask SVDB Directory statement is not specified in the rule file.
Error OPEN1 - problem with access, file type, or file open of file: <i>filename</i> . Compilation of the SVRF rules failed.	One or more of the calibrated rule files (<i>rules.C</i> or <i>rules.R</i>) or the lvs rule file has not been found.
SVRF syntax error on encrypted line <i>linenumber</i> of <i>filename</i> . Compilation of the SVRF rules failed.	One or more of the calibrated rule files (<i>rules.C</i> or <i>rules.R</i>) or the lvs rule file has not been found.
Compilation of the SVRF rules failed.	One or more of the calibrated rule files (<i>rules.C</i> or <i>rules.R</i>) or the lvs rule file has not been found.
Rules file version “vXXXX.X” is not supported by this version of xRC “vXXXX.X”.	Newer versions of the calibrated rule files (<i>rules.C</i> or <i>rules.R</i>) cannot be specified with older release versions of Calibre xRC.
No output files specified for formatter run (PEX NETLIST/REPORT).	A form of PEX NETLIST or PEX REPORT statement is not specified in the rule file.

Table F-1. Calibre xRC Error Messages (cont.)

Message	Possible Causes
Restoration of HDB failed. database does not exist or could not be created.	The PHDB does not exist. The PDB stage was executed before the PHDB was generated with either the -phdb or -lvs command line options.
Nets were not specified for selected net run.	The -pdb -select command line options were specified without the PEX Extract Include statement specified in the rule file.
Bad device pin path “ <i>name</i> ” in PDB model for net “ <i>netname</i> ”. Device will be removed from net model.	This error occurs during the formatter stage. The pdb contains a device with a corrupted pin name.
Analyzer encountered PDB internal error: PDB_DISKFULL.	This error occurs during PDB generation. The location to where the PDB is being written is full.
FATAL ERROR: Gate Layer “ <i>gate_layername</i> ” is not mapped to a calibrated layer.	The <i>gate_layername</i> is not mapped to a calibrated layer. See PEX Map statement for details on how to map the layer. Use the PEX Exception Severity statement to control how this exception is handled during extraction.

Warning Messages

Warning messages should be reviewed to determine if they indicate a real problem in your design.

Table F-2. Calibre xRC Warning Messages

Message	Possible Causes
Layer “ <i>layername</i> ” is not mapped to a calibrated layer.	This warning occurs during the PDB stage. The <i>layername</i> is not mapped to a calibrated layer. See PEX Map statement.
CAPACITANCE ORDER statement is incomplete.	This warning occurs during the PDB stage. The Capacitance Order statement is missing a mapped layer name.
Bad source pin map <i>number</i> . Layout <i>cellname</i> .	This warning occurs during the formatter stage. A <i>filename.stl</i> file has been created for the <i>cellname/hierarchy</i> in a previous run and has missing or added pins compared to the new run.

Table F-2. Calibre xRC Warning Messages (cont.)

Message	Possible Causes
Detected ideal (non-resistive) connection through layer <i>layername</i> .	The warning occurs for each layer that is found to have no assigned or inherited resistivity, but is specified in a Connect statement. It does not list all the derived layers, only layers found while looking at resistance connectivity.
No density window defined for layer <i>layername</i> - switching to nominal thickness.	The warning occurs when the PEX Density Window statement has not been specified. In this case, Calibre xRC uses nominal thickness during extraction.
Detected High Resistance Value on Cell: “ <i>cellname</i> ”, Layer: “ <i>layername</i> ”.	The warning occurs when resistors with values greater than or equal to 1e6 ohms (1Megaohm) are extracted on a layer. To eliminate this warning, make sure that the rules that reference <i>layername</i> in your rule deck are correct.

AMS

Analog mixed signal.

ADvanceMS

A Siemens EDA mixed signal design analysis tool.

backannotation

A process where extracted parasitics are added to the source netlist for parasitic re-simulation.

black box extraction

An extraction method where the contents of cells are ignored during parasitic extraction.

coupled capacitance

The capacitance between two conductors. See [Capacitance Models in Parasitic Extraction](#).

disjoint parasitic

A parasitic element associated with an incompletely routed net.

distributed capacitance

Parasitic capacitance modeled with separate elements distributed over a net that is divided into segments.

domain change (boundary)

In a mixed-signal design, a cell that changes from digital to analog or analog to digital when walking the design top-down.

feedthrough net

A net within an [xcell](#) that connects to a higher-level port outside of the cell but does not connect to any device within.

floating net

A net that is not electrically grounded through connection to a device or xcell port.

floating net coupling

Coupling capacitance between signal nets and floating nets.

formatter

Extraction step which formats the netlist according to command line options and SVRF statements found in the extraction rule file.

fringe capacitance

The capacitance between the side of a conductor and either the substrate (intrinsic) or the bottom or top of another conductor (crossover). See [Capacitance Models in Parasitic Extraction](#).

full hierarchical extraction

A type of hierarchical extraction in which nets are extracted down to user-defined cells, and the contents of the cells are also extracted, preserving hierarchy. See [Hierarchical Memory Extraction](#).

gate-level extraction

A type of hierarchical extraction in which nets are extracted down to user-defined cells, but no further. The PDB and PHDB contain no information about cell contents. See [Gate-Level Extraction](#).

gray box extraction

An extraction method where the parasitic elements between top-level routing cell geometries are included in the extraction. These parasitics are added to the intrinsic capacitance of a top-level net crossing over a cell.

hcell

A user-specified hierarchical cell used by Calibre nmLVS.

hierarchical extraction

A type of extraction which extracts data for each user-defined cell as well as the top level of the design. See [Hierarchical Memory Extraction](#).

in-context cells

Cells that are specified in an xcell file for in-context extraction.

in-context extraction

A type of extraction which extracts the parasitics of a cell with reference to structures outside the cell boundary. The location or “context” of the cell affects the internal parasitics of that cell. See [In-Context Extraction](#).

in-die variation

During parasitic extraction, the drawn dimensions of conductors along with the local density of the material in a region around the conductor are used to determine the actual width, spacing, and thickness of each line.

intrinsic capacitance

The capacitance between a net and substrate (ground). See [Capacitance Models in Parasitic Extraction](#).

lumped capacitance

The amount of parasitic capacitance for a net. The lumped capacitance is represented as a single parasitic capacitor between net and ground and includes all intrinsic and coupled capacitance effects.

map file

A file which maps layout layer names to layer names used in a SVRF rule file.

mixed-signal hierarchical extraction

A type of hierarchical extraction which extracts some user-defined cells in full, and stops at the boundary of others. Contrast full hierarchical extraction and gate-level extraction.

mutual inductance

Defined as the ratio of electromotive force (emf) generated between two inductors, or the full emf effect of one current loop over another.

nearbody capacitance

The capacitance between the sides of two conductors, either on the same layer or different layers. See [Capacitance Models in Parasitic Extraction](#).

net exclusions

Nets in the design for which no parasitic model is extracted.

parasitic models

A set of multi-variable polynomial equations that compute parasitic effects.

parasitic netlist

A netlist containing models of the parasitic effects. The exact format and types of parasitics are specified by SVRF statements and command-line options.

PHDB

The Persistent Hierarchical Database, a database that stores information about your layout.

PDB

The Parasitics Database created by the extraction step. This database contains information about the parasitic capacitance and resistance.

plate capacitance

The capacitance between the lower surface of a conductor and the substrate, or the lower surface of a conductor and the upper surface of another conductor. See [Capacitance Models in Parasitic Extraction](#).

primitive cell

A cell that a designer provides from a standard library (for example, nand, xor, or). In hierarchical extraction, a primitive cell is designated with a -P in the xcell file and does not have parasitics extracted.

probe points

User-specified points on a net that are labeled and used to verify timing.

process corners

The variations on a “typical” process: for instance, metal thickness may not be exactly controlled.

process variation

Deterministic or random variability resulting from manufacturing process steps responsible for creating devices and interconnect in an integrated circuit.

signal net

A net that either has connections to devices or xcell ports, or is designated a port.

replicated device

Devices or cells that are repeated and connected together in a series or parallel combination. Replicated devices may correspond to one device on the source side (netlist or schematic).

self inductance

The change in a magnetic field of a conductor due to a change in current flow.

sensitivity aware

A type of extraction where electronic or physical sensitivities are taken into account during the extraction process. See also process variation.

smashed devices

Devices or cells that consist of drawn layout polygons at the same hierarchical level, also called “flattened”.

source-based extraction

A type of extraction where the extracted parasitics from the layout are included in the source netlist, where layout devices are matched to source devices. See [Backannotating Parasitics to a Source Netlist](#) and [Using the Source Based Flow](#).

SVDB

The Standard Verification Database. The term is also used to indicate the directory named in the MASK SVDB DIRECTORY statement. The SVDB directory also contains the PHDB and PDB.

SVRF rule file

An ASCII file containing Calibre-specific statements. These statements are described in the *Standard Verification Rule Format (SVRF) Manual*.

TICER

Time **C**onstant **E**quilibration **R**eduction method, used to reduce parasitic networks. See [TICER](#).

transistor-level extraction

A type of extraction where the design’s interconnect nets are flattened into a top-level cell. For more information, see [Flat Transistor-Level Extraction](#).

xcell

A user-specified extraction cell. The xcell appears in the generated netlists as a circuit. Every xcell must also be an hcell.

xcell file

An ASCII file that maps xcells to cells defined in the layout. For certain types of extraction, the xcell file settings may also affect whether parasitics are extracted.

— Symbols —

.BIND statements, [74](#)

[], [17](#)

{}, [17](#)

|, [17](#)

— A —

Accuracy trade-offs, [36](#)

adms argument, [157](#)

ADvanceMS defined, [211](#)

AMS defined, [211](#)

AMS extraction

 Spice input, [70](#)

 Verilog input, [70](#)

— B —

bind.inc, [74](#)

Bold words, [17](#)

Boundary defined, [211](#)

— C —

Calibre LVS-H

 circuit comparison, [174](#)

 cross-reference files, [188](#)

 database equivalence, [187](#)

 xcell list, [105](#)

Calibre xRC

 capacitance statements, [175](#)

 ways to run, [17](#)

Calibre xRC, PDB, [157](#)

CALIBRE_HOME variable, setting, [151](#)

Capacitance statements, [175](#)

Checksums, bypassing, [159](#)

CMP data, [112](#)

Command line switches

 -fmt_info, [162](#)

 -fmt_warnings, [162](#)

 -simple, [161](#)

Command reference, [17](#)

Command syntax, [17](#)

Connectivity, [187](#)

Connectivity extraction

 LVS, [174](#)

-corner

 examples, [125](#)

Courier font, [17](#)

— D —

dat files, [187](#)

delay calculator, [70](#)

Design hierarchy, *see* Hierarchy

Device extraction without parasitics, [115](#)

Device recognition

 Calibre LVS-H, [174](#)

Devices

 removing from netlist, [126](#)

Distributed RC netlists

 DSPF output, [191](#)

 example creation, [76](#)

 Spectre output, [191](#)

 SPEF output, [191](#)

Domain change defined, [211](#)

Double pipes, [17](#)

DSPF output files, [191](#)

— E —

Error messages, [207](#)

Errors, [208](#)

Extraction

 devices without parasitics, [115](#)

— F —

Feedthrough nets, hierarchical extraction with, [109](#)

Files

 cross-reference files, [188](#)

 pdb.seg, [186](#)

Floating nets, [109](#)

fmt_info switch

see Command line switches

fmt_warnings switch

see Command line switches

Font conventions, 17

Formatter

.pxi file, 191

.sp file, 191

.sp.pex file, 191

-all switch, 161

-c switch, 161

-r switch, 161

-simple switch, 161

— G —

Gate-level extraction

about, 47

Gray box, 115

Grounds

multiple, 111, 120

— H —

Heavy font, 17

Hierarchy

overview, 47

xcell list, 105

HSPICE output files, 191

— I —

In-die variation

local density, 147

Invocation, CALIBRE_HOME, 151

Isolated devices, 126

Italic font, 17

— L —

Layout netlist, 188

Layout Primary statement

generating PDBs, 187

Licensing information, 17

Local density, 147

Logic gates

gate-level extraction, 47

Lumped C netlists

example creation, 79

LVS connectivity, 174

lvs.rep, 35

lvs.rep.ext file, 35

— M —

messages, 208

Metal fill, 109

Minimum keyword, 17

Multiple ground regions, 111, 120

— N —

Net names

and port names, 131

Netlisting

multiple grounds, 111, 120

Netlists

corners, 126

Nets

pruning devices, 126

— O —

On-chip variation

CMP, 112

process corners, 125

Output files

.sp.pex file, 191

lvs.rep, 35

lvs.rep.ext, 35

name, 192

SPICE, 191

— P —

Parasitic Database (PDB)

segment file, 186

Parasitics to output to RC netlist, 28

Parentheses, 17

pdb.dat files, 187

pdb.seg files, 186

Performance trade-offs, 36

PEX CMP Mode, 112

PEX Ground Layer, 121

PEX Map, 121

PEX Netlist statements

Formatter invocation, 160

formatter output, 192

PEX Probe File, 131

PEX Reduce Parallel, 200

PEX Report statements, 160

Pipes, 17

Praesagus*see* CMP

Primitive cells defined, [213](#)

Probe points
setting, [131](#)

Process variation, CMP, [112](#)

pxi file
purpose, [191](#)

— Q —

Quotation marks, [17](#)

— R —

Reporting
resistor information, [179](#)
Resistor locations, [179](#)

— S —

Segment file, [186](#)
Selected-net flow
PDB, [186](#)
steps, [116](#)
Set probe points, [131](#)
Setting CALIBRE_HOME, [151](#)
Simple output mode, [161](#)
SIMPLE.sdf, [74](#)
SIMPLE.spf, [74](#)
SIMPLE.spf.log, [76](#)
Slanted words, [17](#)
sp files
purpose, [191](#)
sp.pex files
purpose, [191](#)
Spectre
example creation, [79](#)
Spectre output files, [191](#)
SPEF output files, [191](#)
SPICE files
including, [70](#)
SPICE output files, [191](#)
Square parentheses, [17](#)
Standard Verification Database
specifying, [188](#)
Standard Verification Database (SVDB)
.dat files, [187](#)
.pdb files, [186](#)
contents, [188](#)
cross-reference files, [188](#)

layout netlists, [188](#)

Substrate noise analysis, [111](#), [120](#)

SVDB
see also *Standard Verification Database*

SVRF
minimum, [58](#)
SVRF Statements
PEX Reduce Parallel, [200](#)

— T —

Templates
enable, [195](#)
location of, [197](#)
Thickness CMP based, [112](#)
Time-It, [70](#)
Time-it
design flow, [204](#)
manual set, [203](#)
Timing verification, [131](#)
top.dspf, [74](#)
top.inc, [74](#)
Trade-offs, [36](#)

— U —

Underlined words, [17](#)
Usage syntax, [17](#)

— V —

VCMP*see* *CMP*
Verify timing, [131](#)
Verilog translator setup, [70](#)

— W —

Warnings, [208](#)
Ways to run Calibre xRC, [17](#)
Wildcards
xcell lists, [116](#)
Wildcards in xcell lists, [108](#)

— X —

Xcells
wildcards, [108](#)
xcell list, [106](#)

Third-Party Information

Details on open source and third-party software that may be included with this product are available in the *<your_software_installation_location>/legal* directory.

