# Calibre® xRC™ User's Manual

Software Version 2009.1

**Contractor/manufacturer is**:
Mentor Graphics Corporation
8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777.
Telephone: 503.685.7000
Toll-Free Telephone: 800.592.2210
Website: www.mentor.com
SupportNet: supportnet.mentor.com/
Send Feedback on Documentation: supportnet.mentor.com/user/feedback_form.cfm

# Table of Contents

# List of Examples

# List of Figures

# List of Tables

# Chapter 1
# Calibre xRC Overview

This chapter includes the following sections that provide an overview of the Calibre® xRC™ product:



## What is Calibre xRC?

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.

## Why Should I Use Calibre xRC?

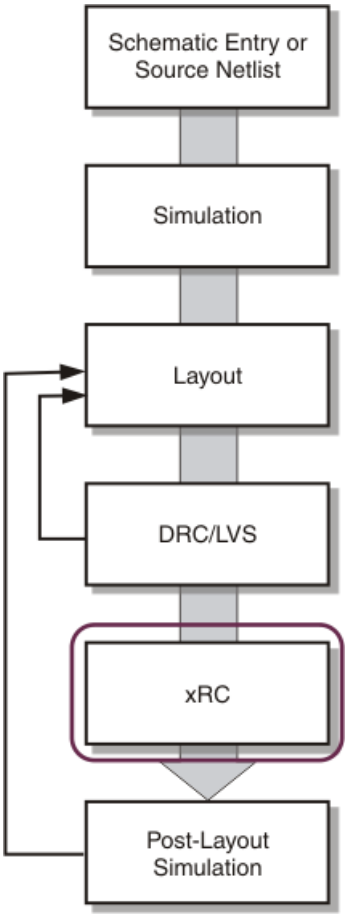
Use Calibre xRC to calculate the parasitics on your design. Parasitic effects can slow down signals, add noise, or cause hot spots in your design, among other problems.

## How does Calibre xRC Fit Into 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**




# What Do I Need Before I Run Calibre xRC?

- 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 calibrexrc, calibrexrcadms, or calibrexrccb 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 Do I Run Calibre xRC?

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

- Interactively from the Calibre® Interactive™ graphical user interface. See Chapter 2, “Getting Started: Parasitic Extraction Using Calibre Interactive”.
- Running it from the command line. See Chapter 3, “Getting Started: Parasitic Extraction Using Calibre Batch Mode”.

# Chapter 2
# Getting Started: Parasitic Extraction Using 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 properly set up.

# 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
- calinteractive and calibrexrc licenses available
- Either the $MGC_HOME or $CALIBRE_HOME environment variables defined

Calibre tools require that the CALIBRE_HOME environment varialbe be set. See "Setting the CALIBRE_HOME Environment Variable" in the *Calibre Administrator's Guide* for details.

## Methods

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 **Tools** > **Calibre Interactive** > **Run PEX** |
| Unix 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 your different types of extractions.

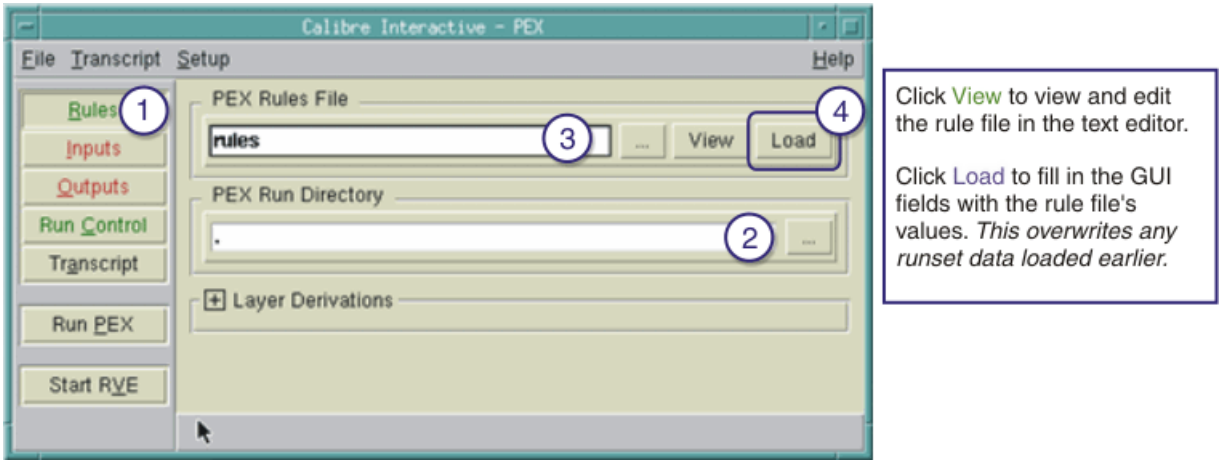
To load a runset, use the file browser by clicking the Browse (**…**) button. After navigating to the runset, select it and click **Open**. In the main dialog, click **OK**.

To skip loading a runset, click **Cancel**.

# Specifying Rule File for PEX Run

1. Click **Rules**.
2. Specify the run directory name. You can use the Browse (**…**) button to select the run directory name from a list.
3. Specify the rule filename. You can use the Browse button to select the rule filename from a list. You can 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**: 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 *schematic* netlist or source files, used when output should have schematic's information. (See "Using Schematic Netlist Input in the Extracted Netlist (Optional)" on page 23.) | 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)" on page 23.) | No |
| Blocks | Specify blocks for xRC-ADMS extraction (See "Running ADMS Extraction" on page 70 for information on ADMS extraction.) | No |
| Probes | Specify probe points. | No |

## Defining Input Data Names in the Extracted Netlist

1. Click **Inputs**.
2. Choose **Layout**.
3. Select a format type using the Format dropdown menu.
4. 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 **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.
5. Specify the layout top cell name.

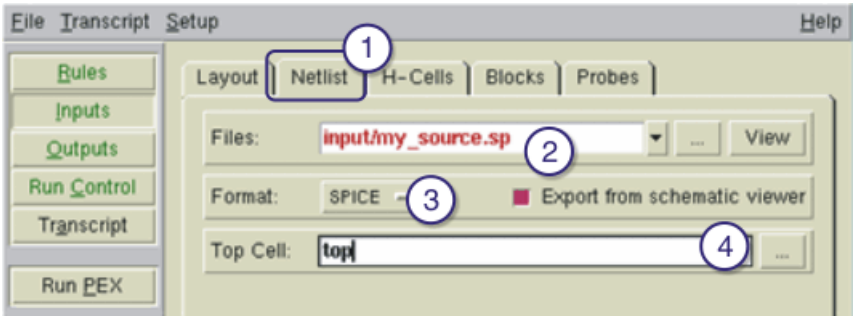
## Using Schematic Netlist Input in the Extracted Netlist (Optional)

If you want to use source names instead of layout names in the extracted netlist:

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

   If you are working also 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**



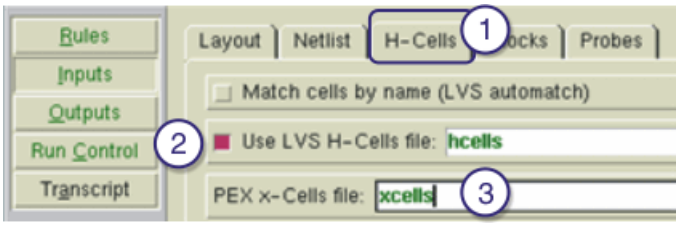

## Defining H-Cells Input (Gate-Level, Hierarchical Extraction and ADMS Only)

**Note**

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

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 -lvs" step uses the hcell file; the "Calibre -xrc" steps use the xcell file. You can use the same file for both Calibre LVS and Calibre xRC.
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.

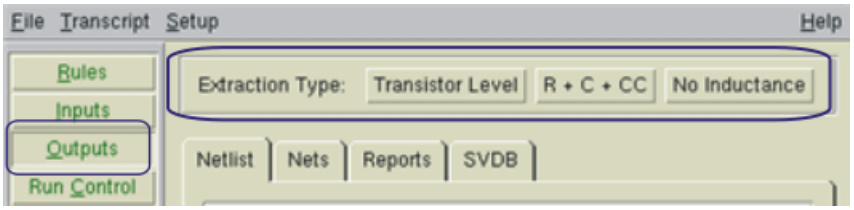
**Table 2-3. Tabs in Outputs Pane**

| Tab | Purpose | Required |
|---|---|---|
| (no tab) | Specify extraction type. (See "Defining the Parasitic Netlist".) | Yes |
| Netlist | Specify the type of netlist. (See "Specify the Netlist" on page 25.) | Yes |
| Nets | Restrict the extraction to only some nets. (See "Restrict the Nets (Optional)" on page 26.) | No |
| Reports | Set up LVS and xRC reports. (See "Set Up Reports (Optional)" on page 26.) | No |
| SVDB | Add information types to the SVDB. (See "Add to the SVDB (Optional)" on page 27.) | No |

## Defining the Parasitic Netlist

1. Click **Outputs**.

**Figure 2-4. Extraction Type Settings**




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

   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.

ADMS extraction is similar to gate-level extraction. It provides additional automation for integrating analog and digital blocks for simulation and requires an additional license, calibrexrcadms.

See the "Types of Extraction" chapter for more details.

3. 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.

4. Select the Inductance option to extract self-inductance and mutual-inductance parasitics. This requires an additional license, calibrexl, to run. Inductance extraction is covered in detail in the *Calibre xL User's Manual*.

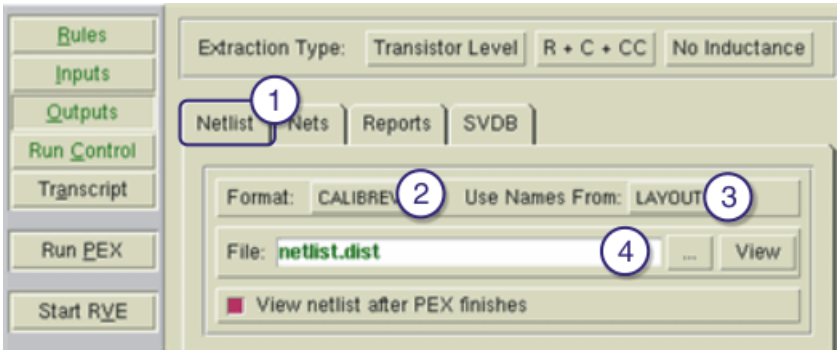
**Note**

The second button on the Extraction Type: line controls the information that is extracted into the PDB. The **Setup > PEX Options** panel 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

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 2-5. Describing the Output Format**




## Restrict the Nets (Optional)

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.

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**: 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)

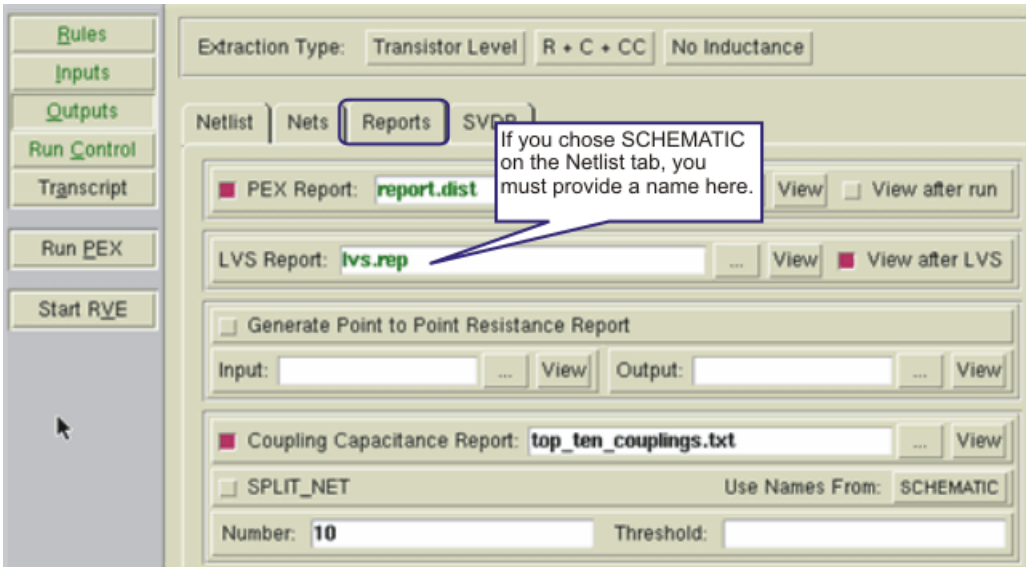
1. Click the **Reports** tab.
2. Choose report options and specify filenames as appropriate. You must specify an LVS report name if you chose SCHEMATIC on the **Netlist** tab, Use Names From: dropdown list.

There are four 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” on page 101.
- 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" on page 97.

**Figure 2-6. Enabling Report Output**




## Add to the SVDB (Optional)

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

   The Generate cross-reference data for RVE option (on by default) generates the binary cross-reference database (XDB) for RVE and the query server.
3. If you need to start the 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.

# Running 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.

# Using Calibre RVE to Highlight Parasitics in the Layout

To highlight parasitics in the layout, you must have the layout editor open and have started Calibre Interactive from it.

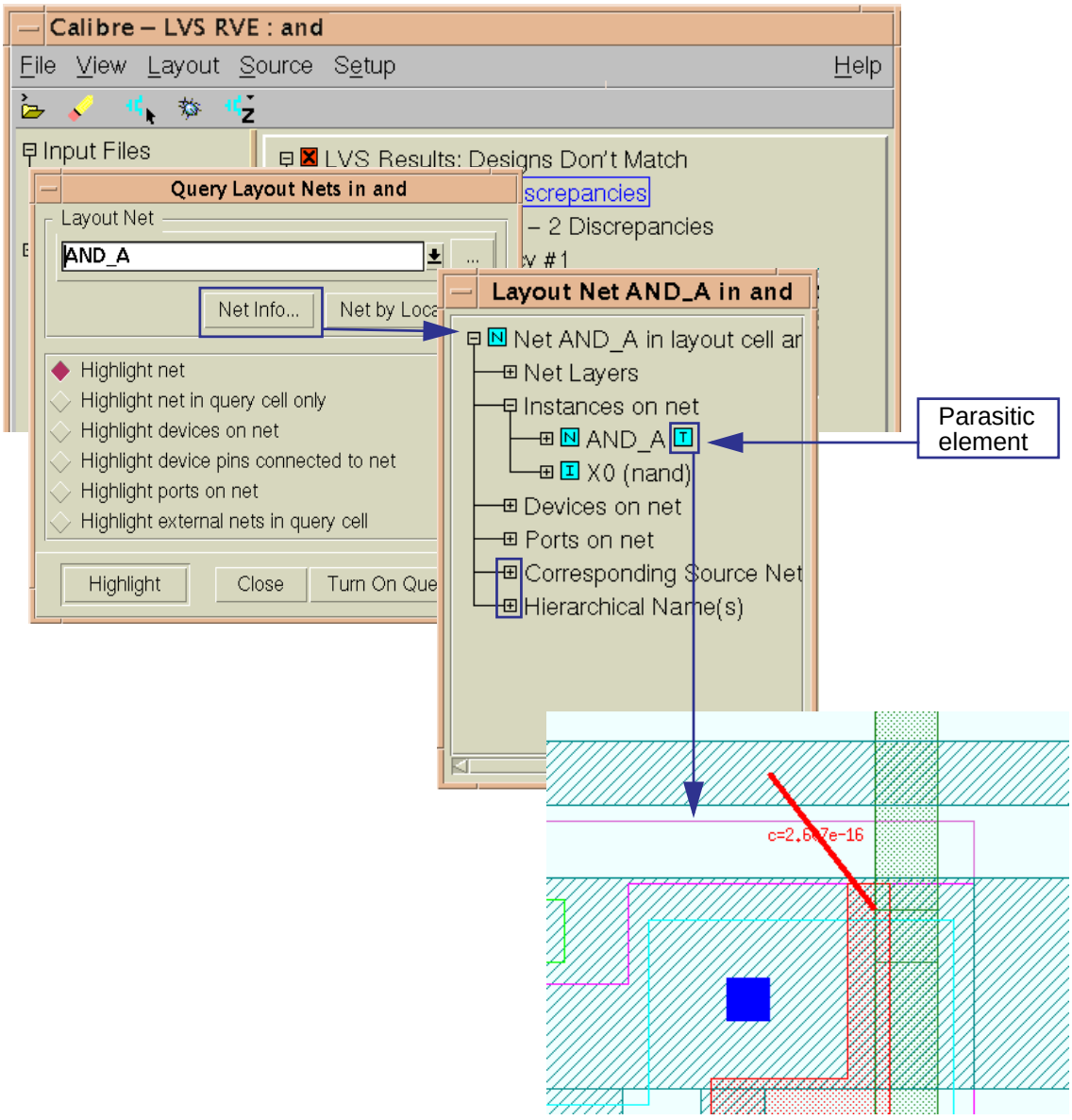
1. Wait for the parasitic extraction to finish. When the transcript pane shows

   ```
   --- CALIBRE xRC::FORMATTER COMPLETED
   ```

   all stages have finished.
2. If RVE did not open automatically, click **Start RVE** in the Calibre Interactive window.
3. In RVE, select **Layout > Net Queries**.
4. Specify the Layout Net name in the Query Layout Data window.
5. Click the **Net Info** button in the Query Layout Data window.
6. Click on plus signs in the Net ENH window to expand parasitics tree.
7. Click on a parasitic element to highlight the element in the layout.

   See Figure 2-7 on page 29.

**Figure 2-7. Querying Nets with RVE**



# 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 the steps below to add the PEX options panel to the GUI:

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.

Here are some of the capabilities:

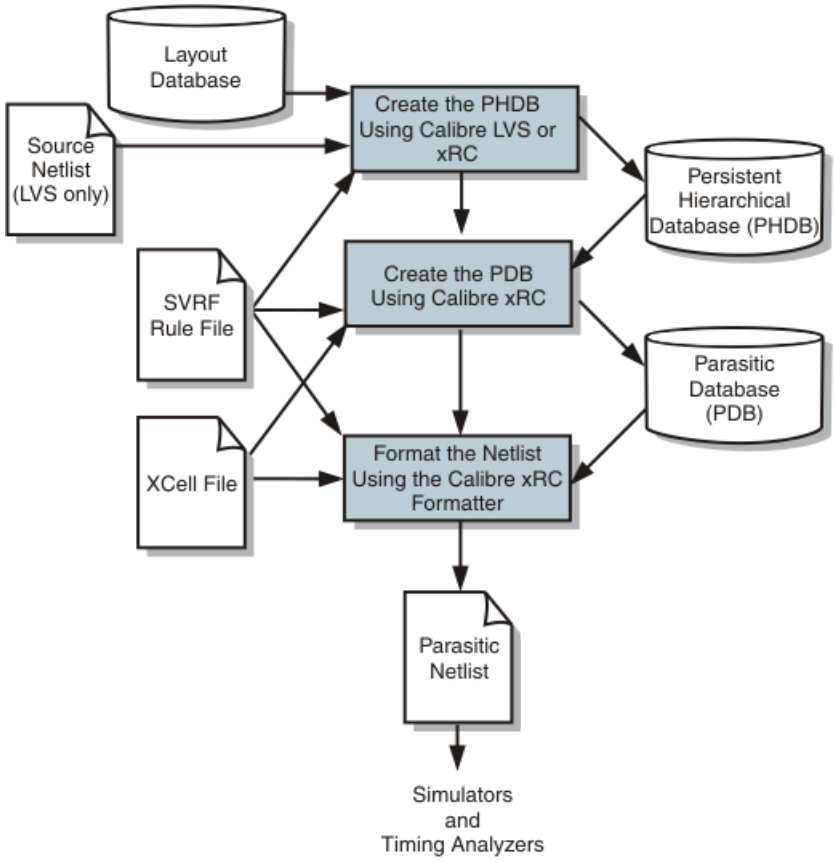
**Table 2-4. Adjustments Available in PEX Options**

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

Figure 3-1 shows the three steps of the overall extraction flow. Each step has many possible customizations based on your particular needs.

**Figure 3-1. Calibre xRC Extraction**



# Before You Begin

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

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.

**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 xRC.

## Procedure

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

   a. 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, and LEF/DEF formats.

   b. If you require source or schematic names in the generated netlist, 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.

**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.

- **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.

## Task Validation

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. The file is not used in parasitic extraction.
- **LVS Extraction file (default: lvs.rep.ext)** — If you used Calibre nmLVS, it will write the results of the circuit extraction to this file. The file is not used in parasitic extraction.
- **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)" on page 256
- "calibre -lvs" on page 167
- "calibre -xrc -phdb" on page 168

# 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.

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.

  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.

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

  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.

## 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 "Creating an Xcell List" on page 122 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 would perform 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 PDB Step**

| Required | Extraction Options[1] | Parasitic Options[1] | Additional Options | Required |
|---|---|---|---|---|
| calibre -xrc -pdb | no switch (flat)<br>-xcell ***xcell_file***<br>-xcell ***xcell_file*** -full<br>-xcell ***xcell_file*** -incontext | -c<br>-r<br>-rc<br>-rcc<br>-adms | -cb<br>-asic / -noasic<br>-select / -cselect<br>-64 | ***rule_file*** |

1. Choose only one from this column.

The procedures in Chapter 6, "Basic Extraction Methods" give complete command line invocations.

## Task Validation

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

- For details on invocation options, "calibre -xrc -pdb" on page 170.
- For a detailed comparison of the different extraction methods, see Chapter 4, "Types of Extraction".
- For an explanation of the types of parasitic network, see Chapter 5, "Producing Parasitic Models".
- For more information on the xcell file, see "Controlling Hierarchy with Xcells" on page 121.
- For information on extracting parasitic inductance, see the *Calibre xL User's Manual*.

# 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.

  The PHDB contains connectivity information.

- **PDB** created in step 2.

  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.

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 *process corners* defined in the rule file. |
| -adms | Produces netlists suitable for Mentor Graphics 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.

## Task Validation

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

- For details on invocation options, see “calibre -xrc -fmt” on page 173.
- For more information on modifying the output, see Chapter 10, “Controlling Netlisting”.
- For files produced by type of output, see “Netlists” on page 260.

# Chapter 4
# Types of Extraction

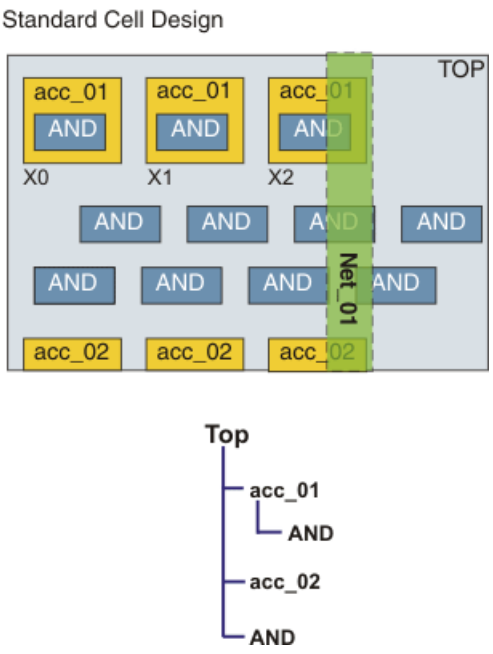
The Calibre xRC extraction engine is able to extract interconnect parasitics hierarchically. Hierarchical extraction is usually faster and requires less memory; the hierarchical netlists it creates are also smaller than equivalent non-hierarchical netlists and easier to simulate. This chapter provides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



## About the Example Design

Figure 4-1 shows the design layout used to illustrate the output of the extraction types. The grey box represents a cell boundary. Inside the cell are two complex cells, acc_01 and acc_02, and a simple logic gate AND. The glue logic and interconnects are *not* shown. Passing over the cell, but not part of it, is a large interconnect labelled Net_01.

**Figure 4-1. Example Design**

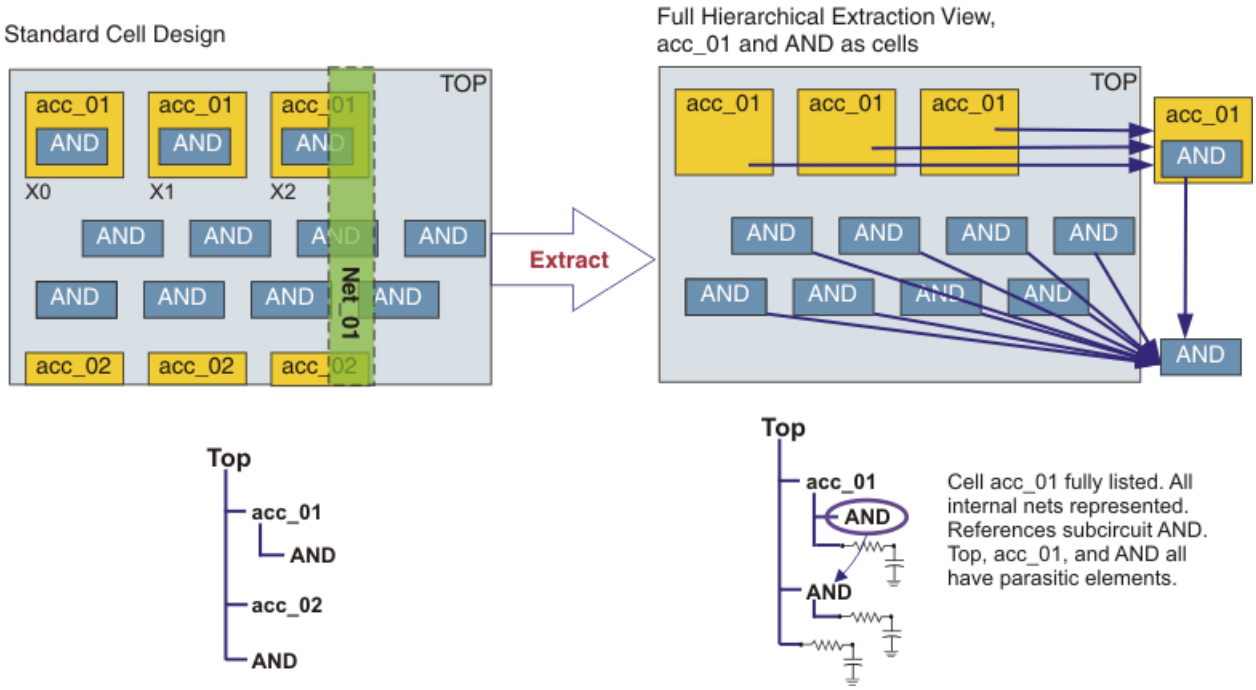


# Full Hierarchical Extraction

Full hierarchical 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, as shown in Figure 4-2, and models each net in the xcell. The fully-hierarchical netlist is suitable for use with a hierarchical simulator.

**Figure 4-2. Full Hierarchical Extraction**




In contrast to gate-level extraction, where the output includes netlist data only to the level of the cell's boundary, hierarchical 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**

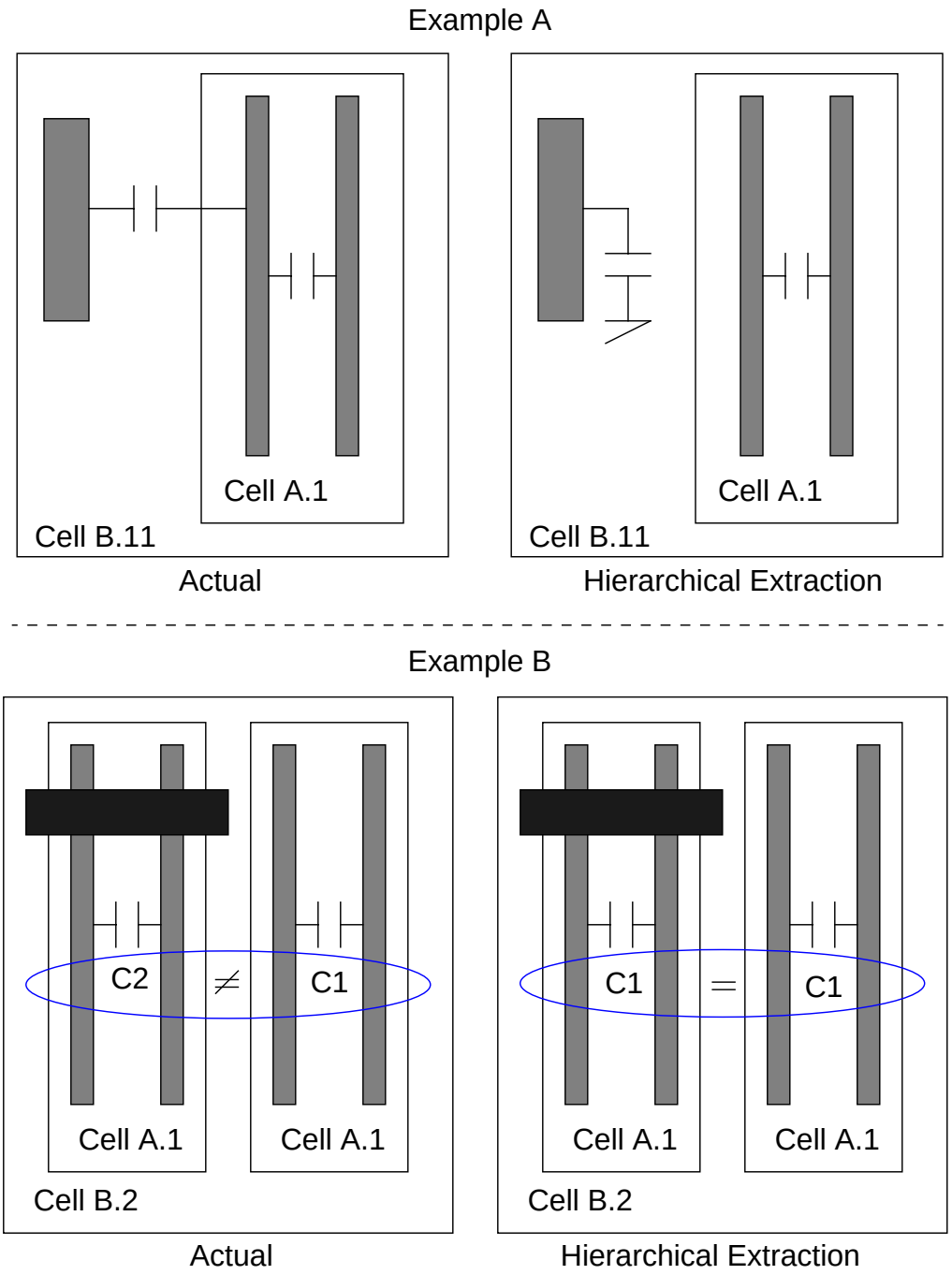
Hierarchical extraction optimizes extraction runtimes and netlist size. The actual capacitance values for a net extracted hierarchically and flat will be slightly different.

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.

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



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.

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

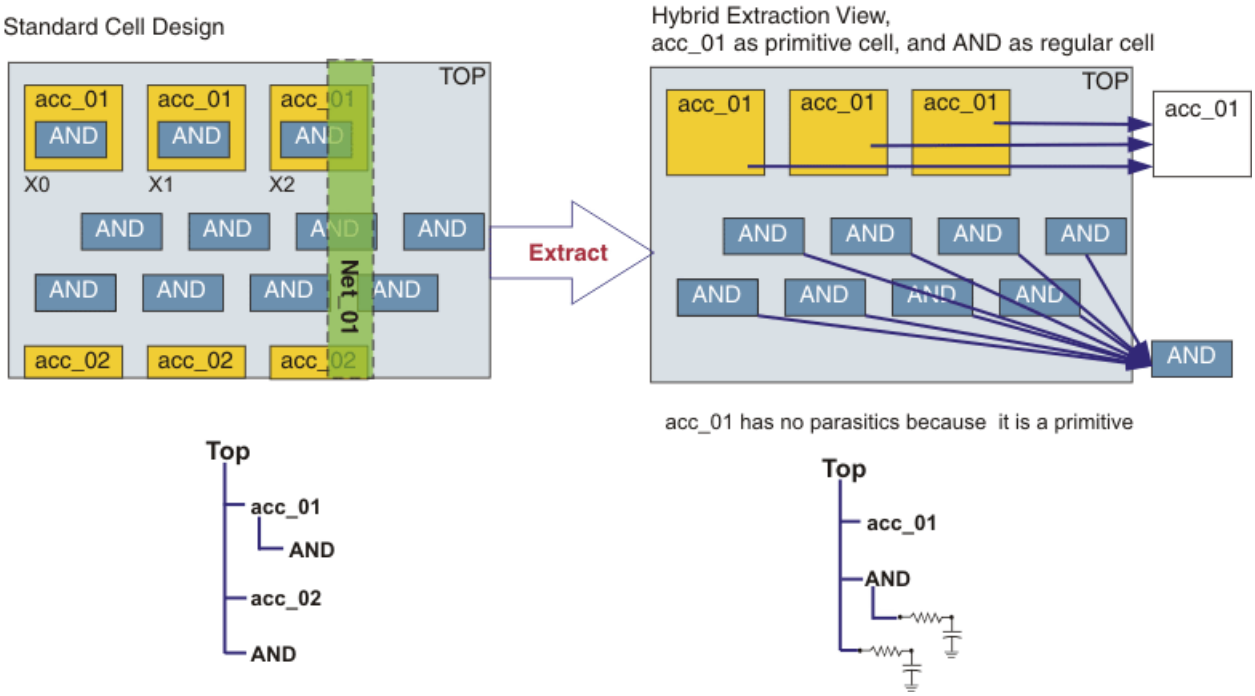
- “Running Full Hierarchical and Hybrid Extraction” on page 66

# Hybrid Extraction

Hybrid extraction combines cell hierarchy and primitives, making it useful for mixed-signal designs. Primitives are identified by -P flags in the xcell list, making hybrid extraction possible with either gate-level or full hierarchical extraction. It is identified as a separate extraction type because you will need to treat the netlists created this way differently.

As shown in Figure 4-4, primitive cells do not contain any information. The parent cell references the primitive cell, but you must provide a model for simulation.

**Figure 4-4. Hybrid (Primitives and Hierarchical) Extraction**




The hybrid format differentiates between two cases:

- Primitive cells whose internals will not be extracted. These are typically standard cells whose contents have been modeled separately. Interconnect is extracted to the boundary of the cell as with gate-level extraction.
- Non-primitive cells whose internals will be extracted if -full was used. These cells are extracted out of context for improved extraction performance or for the purpose of maintaining hierarchy at the cost of some capacitance accuracy.

In both cases the netlist preserves a level of hierarchy for the listed cell. For more information on the xcell file, see "Controlling Hierarchy with Xcells" on page 121.

## Related Topics

- "Running Full Hierarchical and Hybrid Extraction" on page 66

# 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 ***xcell_file*** | Regular gate-level or hybrid extraction. Context IDs are ignored. |
| -xcell ***xcell_file*** -incontext | Only the cells specified with context IDs are extracted and any cells they reference that are identified in the xcell list. *The primary cell is not extracted.* See Figure 4-5 on page 45.<br><br>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 ***xcell_file*** -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. See Figure 4-6 on page 45. |

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.

Figure 4-5 and Figure 4-6 on page 45 contrast invoking with "-xcell -incontext" and "-xcell -full". In both cases, the xcell file is the same.

**Figure 4-5. In-Context Extraction (-xcell -incontext)**

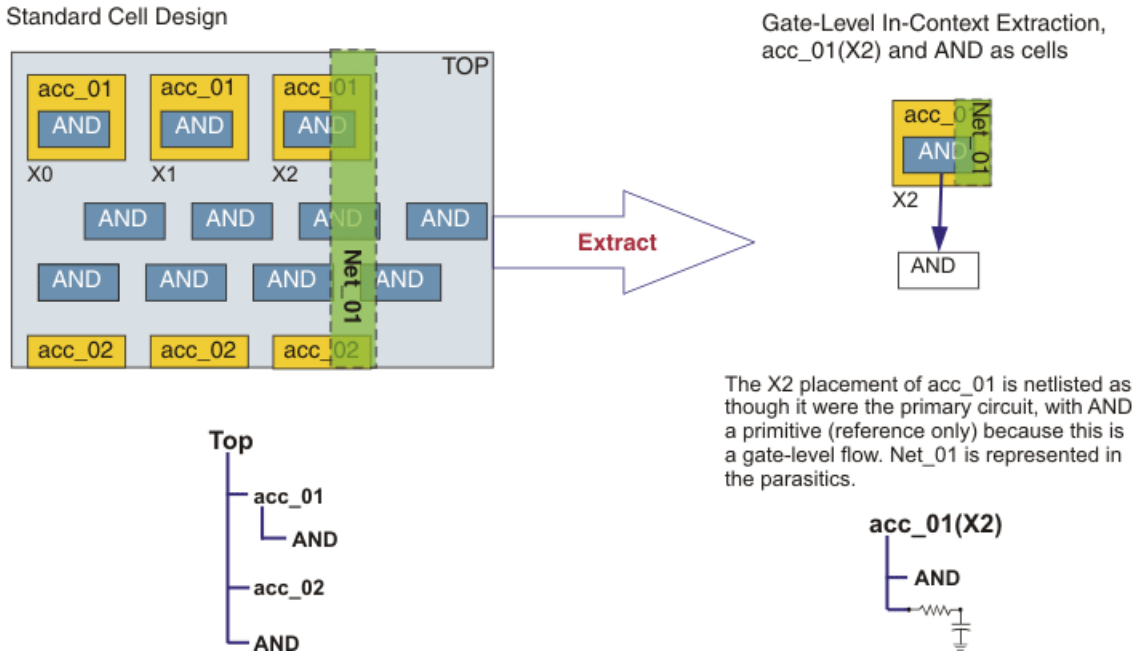
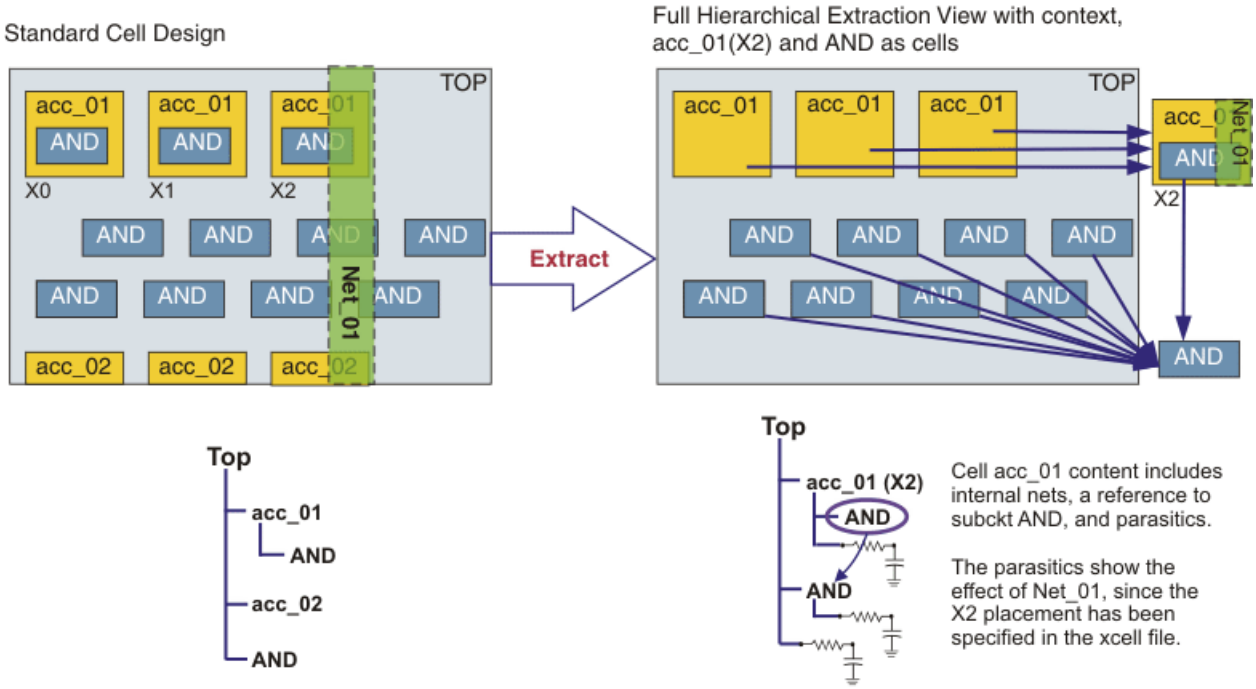



**Figure 4-6. In-Context Extraction (-xcell -full)**




## Related Topics

- "Discovering Layout Paths for In-Context Cells" on page 123
- "Extracting A Placed Cell" on page 103

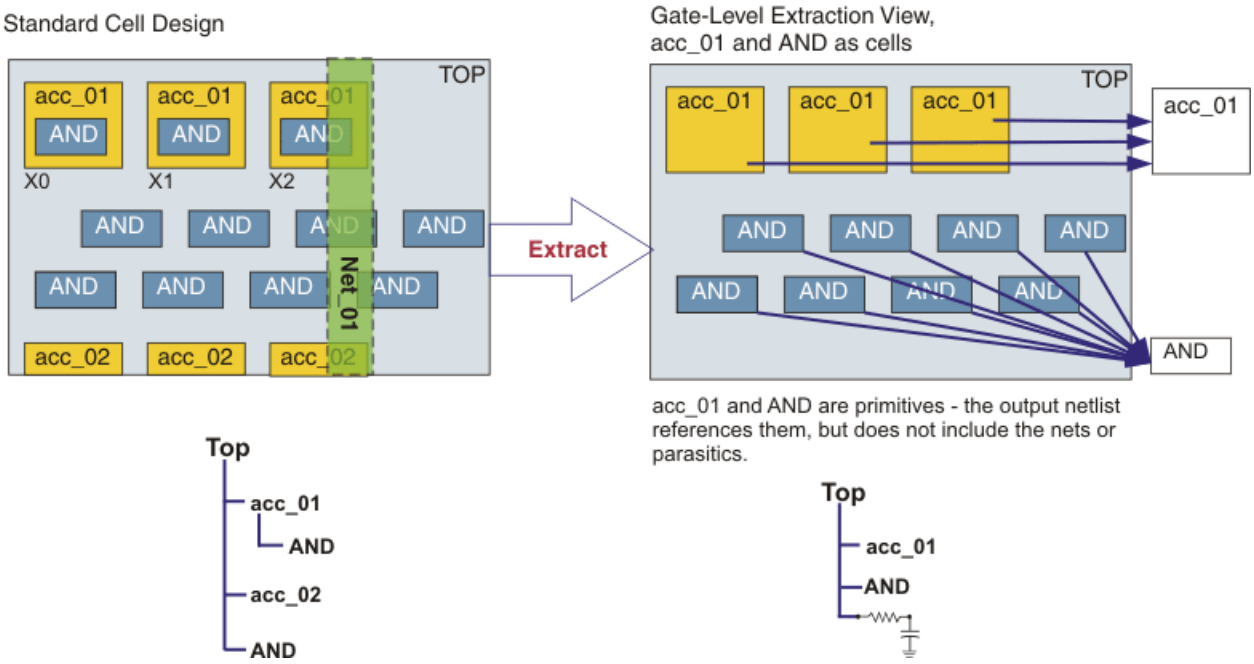
# Gate-Level Extraction

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.

**Figure 4-7. Gate-Level Extraction**




The highest hierarchical xcell instances define the hierarchy. In gate-level extraction xcell instances do not contain other xcell instances. Notice in Figure 4-7 how the acc_01 cell does not include the AND, even though AND is also a cell. 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

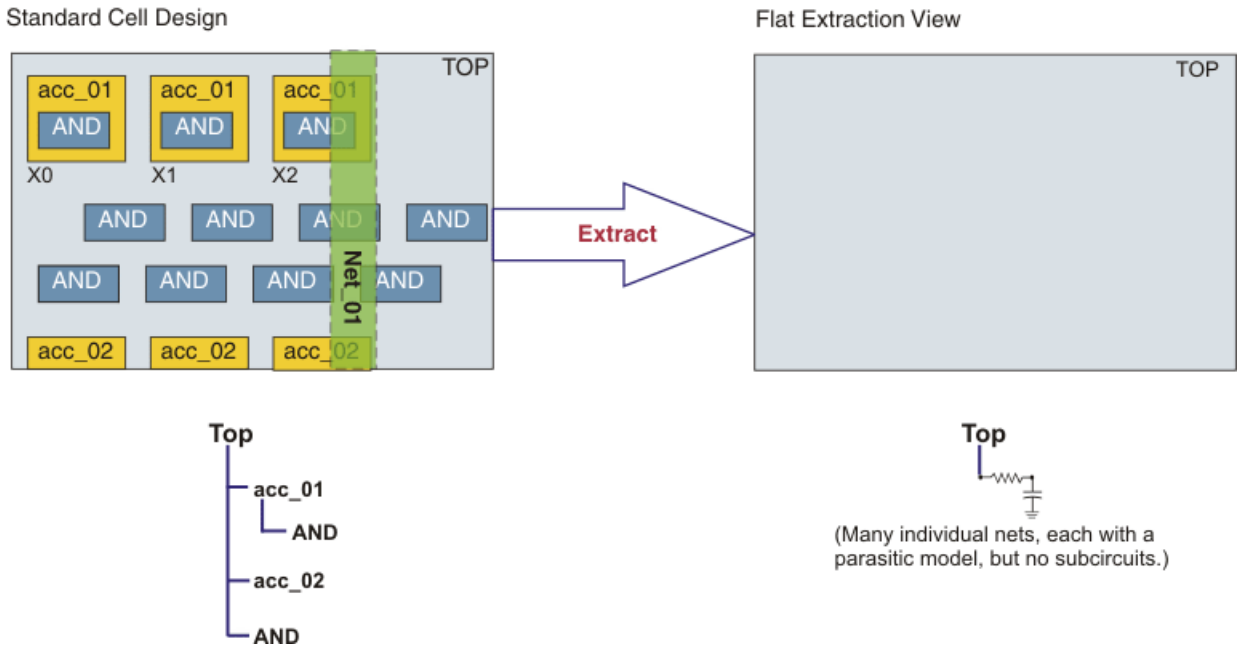
- "Running Gate-Level Extraction" on page 62

# 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.

**Figure 4-8. Flat Transistor Extraction**




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.

## Related Topics

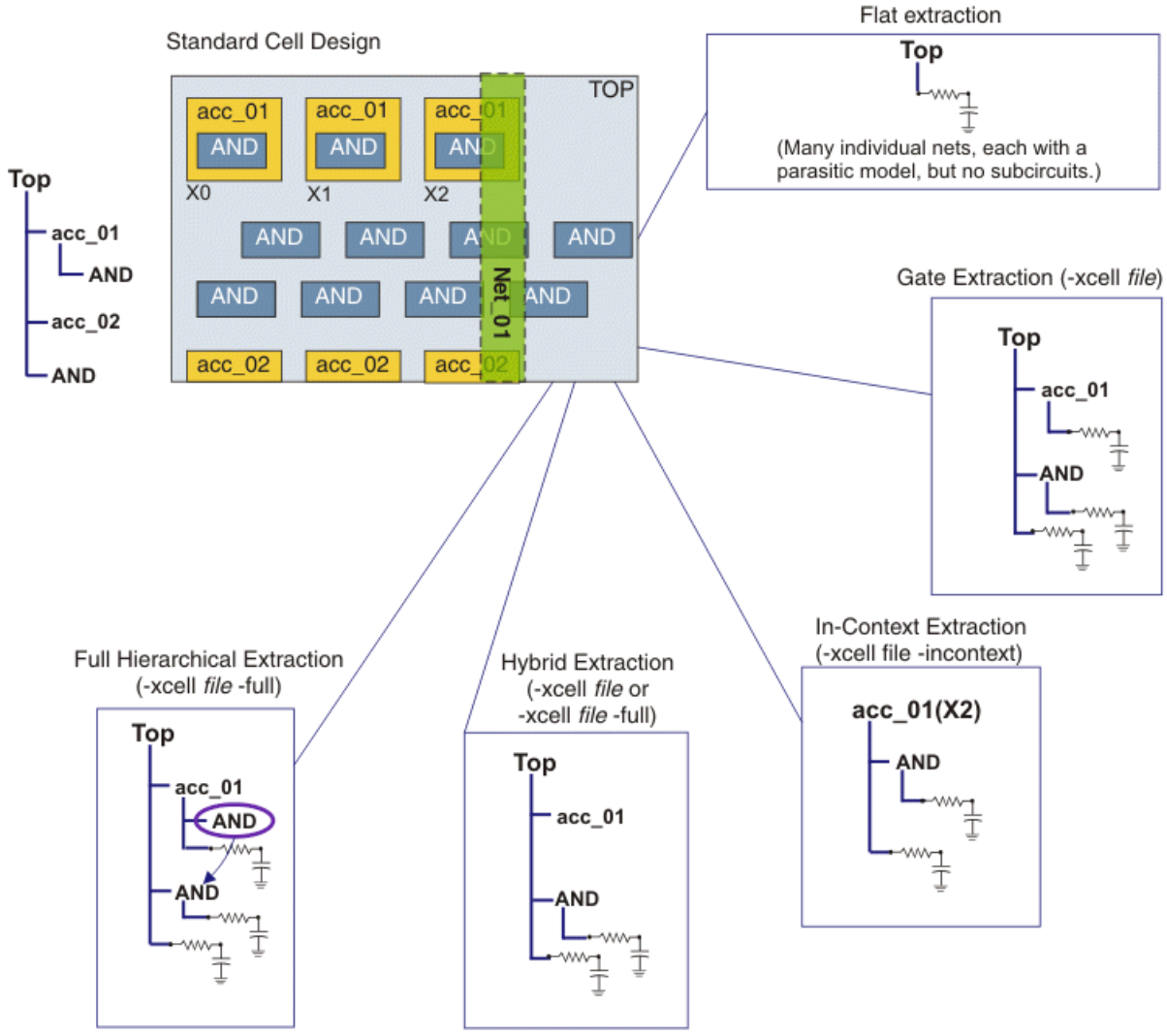
- "Running Transistor-Level Extraction" on page 59

# Comparison of Extraction Types

As shown in Figure 4-9, 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 hybrid 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.

**Figure 4-9. From One Design, Five Levels of Simulation Detail**




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 hybrid extraction to get compact, detailed models on some areas, and a reference that you can fill in with a simplified model in others.

# Chapter 5
# Producing Parasitic Models

This chapter includes the following sections that provide information for producing parasitic models. For a more general discussion of parasitic effects, see Appendix A, "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**

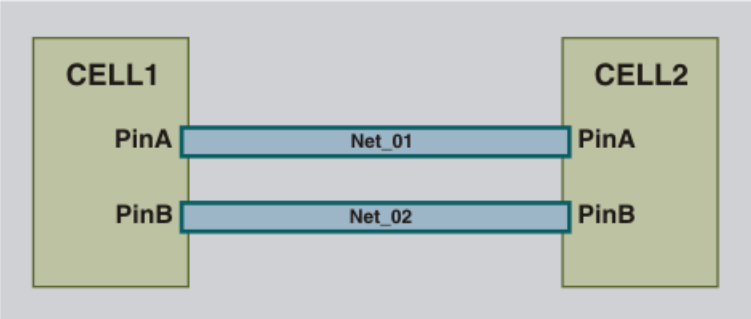



Figure 5-2 shows how the capacitors are represented in the model when the -g command line option is used in the formatting step:

```
%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 Value Added to Intrinsic Capacitors**

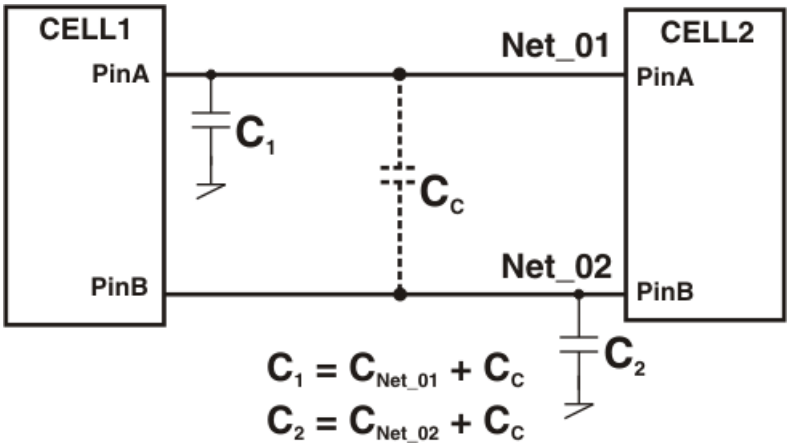
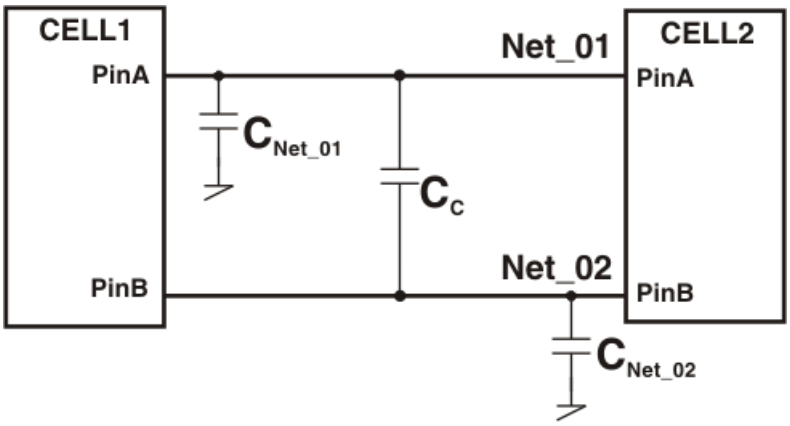



Figure 5-3 shows how the extracted capacitors are represented in the model without the -g command line option:

```
%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**

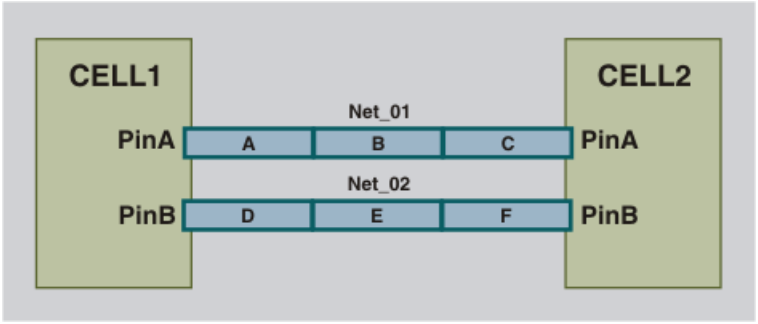
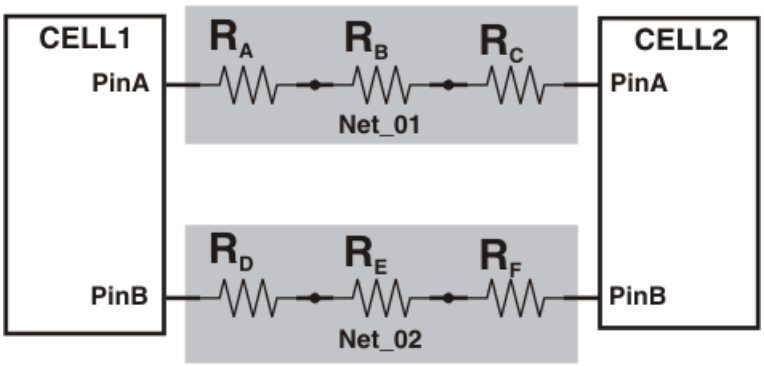



Figure 5-5 shows the extracted net model for distributed R, using the following command lines:

```
%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**

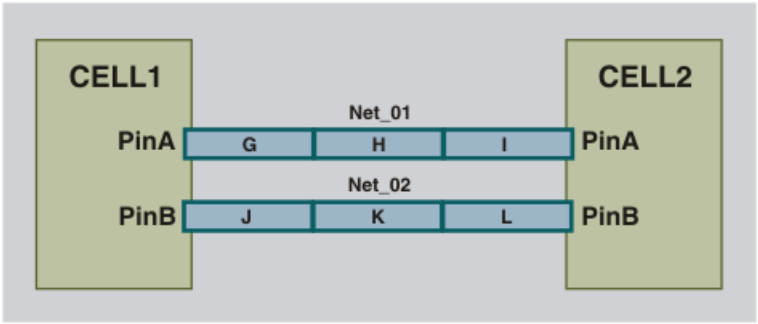


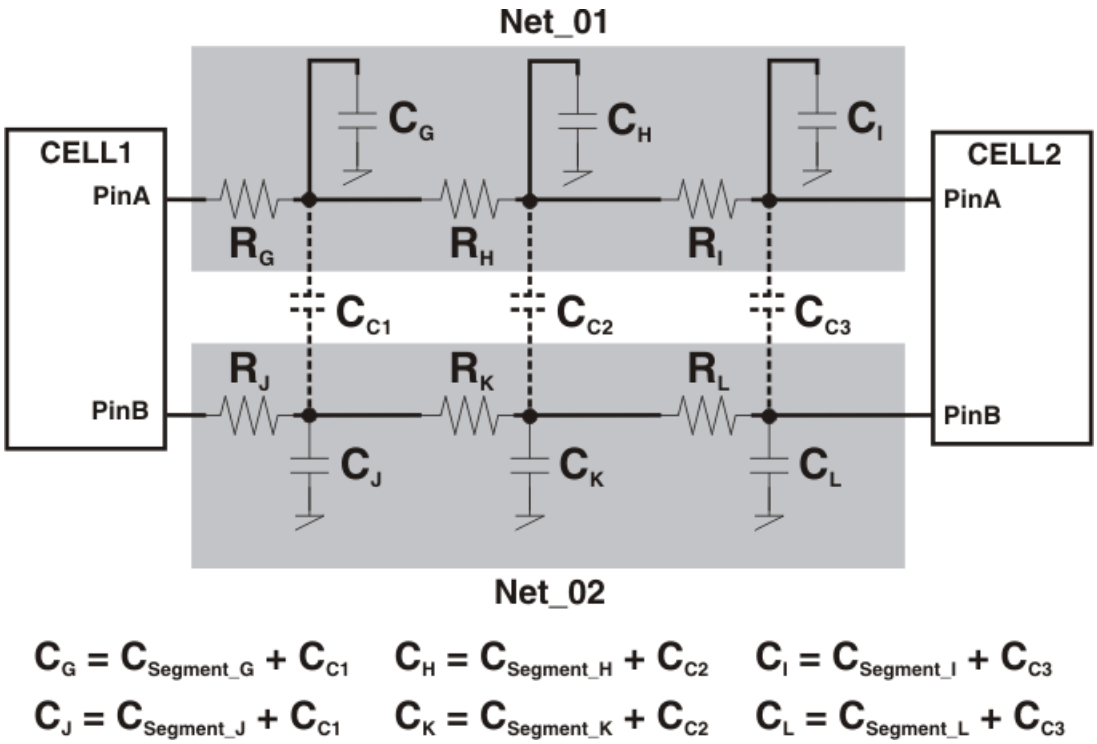

Figure 5-7 shows the extracted net model for distributed R and C, using the following command lines:

```
% calibre -xrc -pdb -rcc
% calibre -xrc -fmt -g [ -rcc | -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 $C_G$ 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**




## 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**

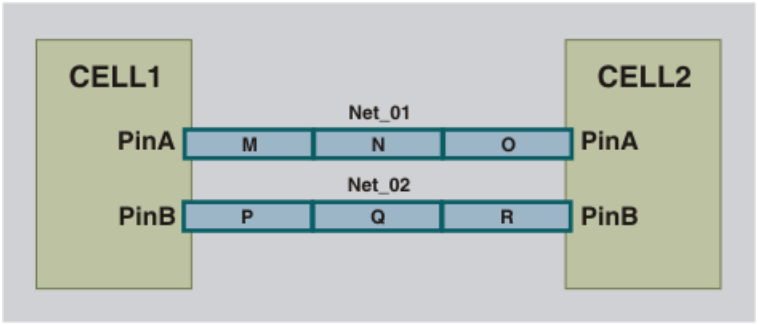


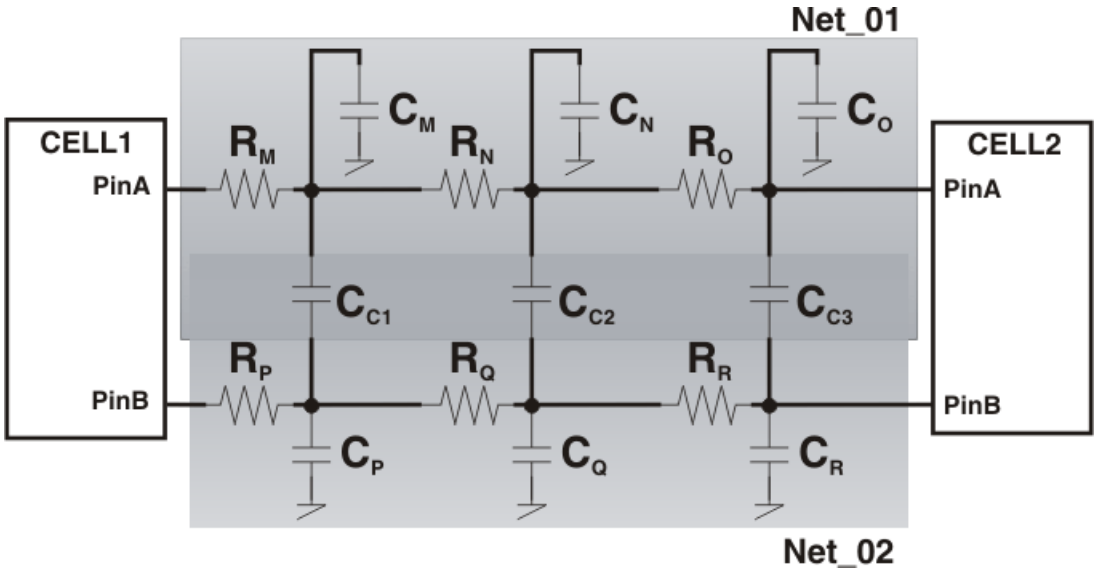

Figure 5-9 shows the extracted net model for distributed R with coupled C when you use the following commands:

```
% 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 capacitvely coupled through $C_{C1}$ to $C_{C3}$. 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.



## By Extraction Type



## By Netlist Type



## By Report



## By Design Hierarchy

# 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 calibrexrc or calibrexrccb license.
- SVRF file for the type of parasitic 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.

  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

Transistor-level extraction is also known as “flat” 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 Flat Netlist from the Command Line

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

### Requirements

- 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_flag rules
   ```

   where ***parasitic_flag*** 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_flag*** 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
   ```

### Task Validation

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 flat 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" in Output Reference

# Creating a Flat Netlist from Calibre Interactive

This procedure only specifies the settings particular to running transistor-level extraction.

## Requirements

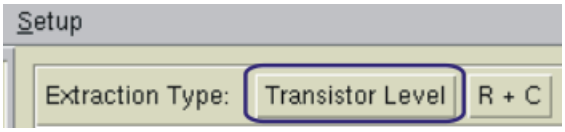
- Layout database that is LVS-clean.
- A valid PEX rule file for this layout.
- For more information refer to Prerequisites for Performing Parasitic Extraction.

## Procedure

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

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

**Figure 6-1. Transistor Level Setting**




4. 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.

## Task Validation

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

### Requirements

- 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

Flags that are specific to gate-level extraction are in bold.

1. 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
```

2. Extract parasitic effects.

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

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

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

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

## Task Validation

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 flat 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" in Output Reference

"Gate-Level Extraction" in Types of Extraction

# Creating a Gate-Level Netlist from Calibre Interactive

This procedure only specifies the settings particular to running gate-level extraction.

## Requirements

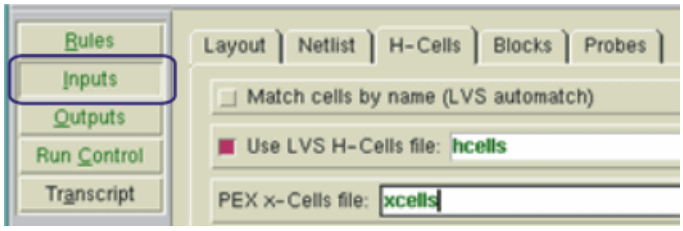
- Hcell file or Hcell statement that includes all cells also listed in the xcell file.
- Xcell file listing cells which will not undergo parasitic extraction. All entries are treated as primitives. You can re-use the hcell file.
- Layout database that is LVS-clean.
- A valid PEX rule file for this layout.

  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 6-2. Specifying HCell and XCell Files in Calibre Interactive**




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

**Figure 6-3. Gate Level Setting**


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

## Task Validation

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" in Types of Extraction

# Running Full Hierarchical and Hybrid Extraction

The difference between full hierarchical extraction and hybrid 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

### Requirements

- 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

Flags that are specific to hierarchical extraction are in bold.

1. If you are netlisting to DSPF, set the PEX_NASSDA environment variable. It improves modeling of feedthrough nets, but only affects DSPF output.
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 -full -parasitic_flag rules
```

where *parasitic_flag* 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
```

## Task Validation

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" in Output Reference

"Full Hierarchical Extraction" and "Hybrid Extraction" in Types of 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.

## Requirements

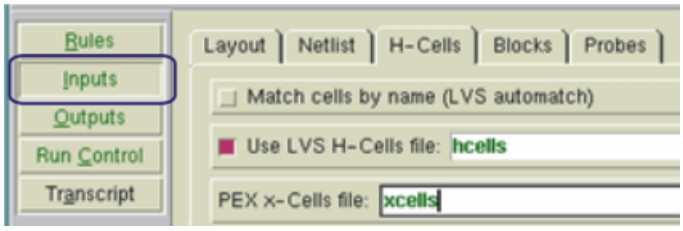
- A valid PEX rule file for this layout. If you are creating a DSPF netlist for use in a hierarchical simulator that accepts a position file, add PEX Netlist Position File.
- Hcell file or Hcell statement that includes all cells also listed in the xcell file.
- Xcell file listing cells to preserve in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

  For more information refer to Prerequisites for Performing Parasitic Extraction.

## Procedure

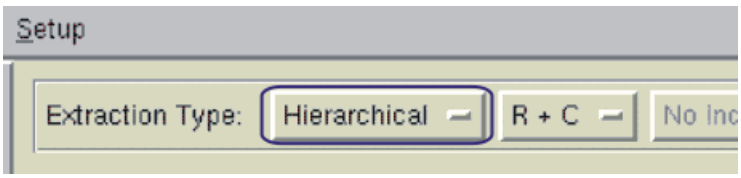
1. Start the PEX interface in Calibre Interactive:
2. Load a runset or rulefile.
3. Specify the hcells, xcells, and extraction type.
   a. Click the **Inputs** button in the left pane. Select the **H-Cells** tab. Specify the hcell and xcell files in the appropriate fields. (They can be the same file.)

**Figure 6-4. Inputs Pane for Hierarchical Extraction**




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

**Figure 6-5. Outputs Pane for Hierarchical Extraction.**




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

## Task Validation

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

"Full Hierarchical Extraction" in Types of Extraction

# Running ADMS Extraction

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

## 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*.

### Requirements

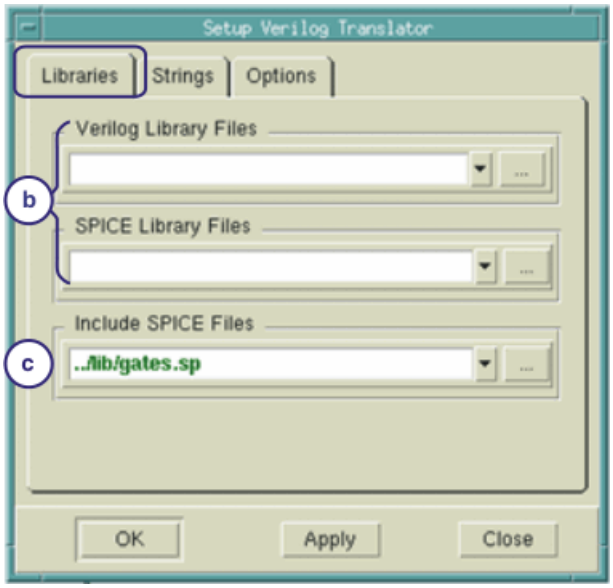
- A calibrexrcadms license in addition to the calibrexrc or calibrexrccb license.
- A valid PEX rule file for this layout.
- Hcell file or Hcell statement that includes all cells also listed in the xcell file.
- Xcell file listing primitive cells.
- Verilog libraries for the design source.
- A delay calculator, such as Mentor Graphics 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 on the next page.
   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 6-6. Setup Verilog Translator**



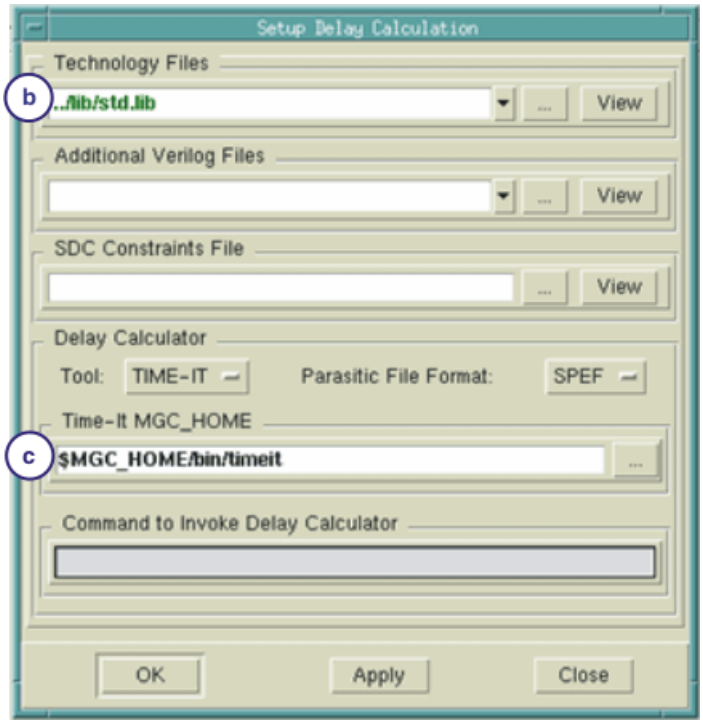

For information on other options in the Verilog Translator, see "Setting Up the Verilog Translator (v2lvs)" in the *Calibre Interactive User's Manual*.

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 on the next page.
   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 SupportNet.

**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 6-7. 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.

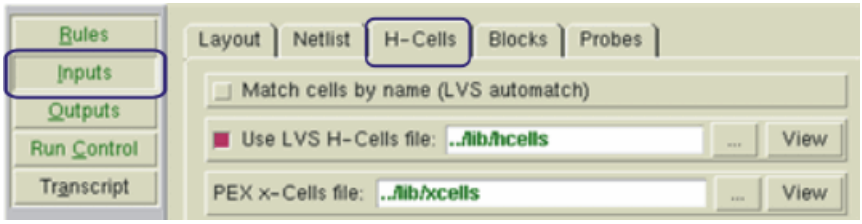
   b. Enter the names of all the source netlist files for LVS and extraction. If a file includes other files, the other files do not need to be separately listed.

   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 6-8. Netlist Tab for ADMS**

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

**Figure 6-9. H-Cells Tab for ADMS**

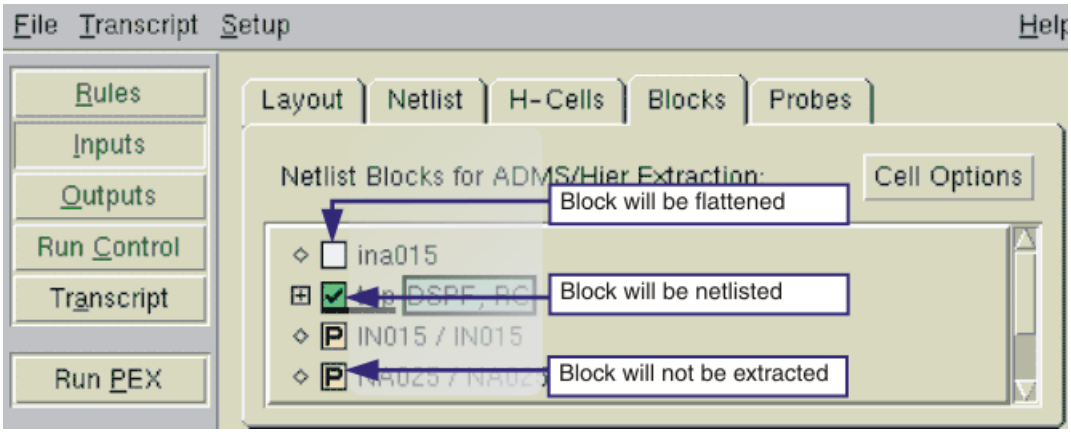



8. Fill out the **Blocks** tab. Here you specify the blocks to be netlisted and the output format to use. The blocks are identified from the source files in the Netlist tab. The xcell file determines whether blocks are primitives.

   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.

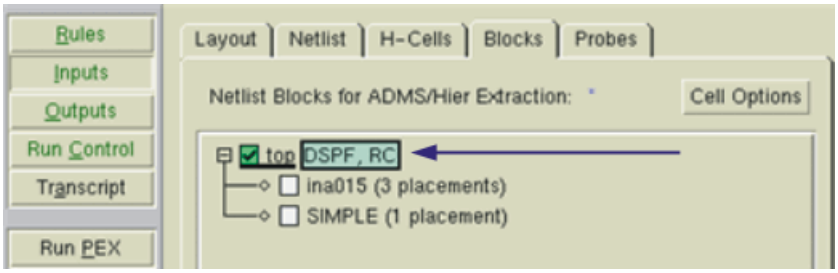
      You cannot unselect the top cell; it is always netlisted. Also, you cannot change primitives, marked with a P, from the interface.

**Figure 6-10. 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 below.

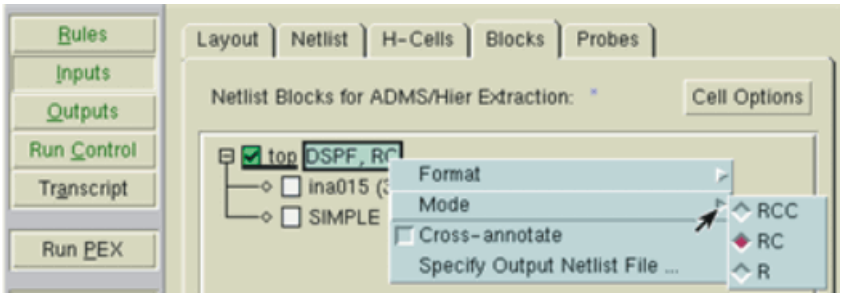
**Figure 6-11. Formatted Block**




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.

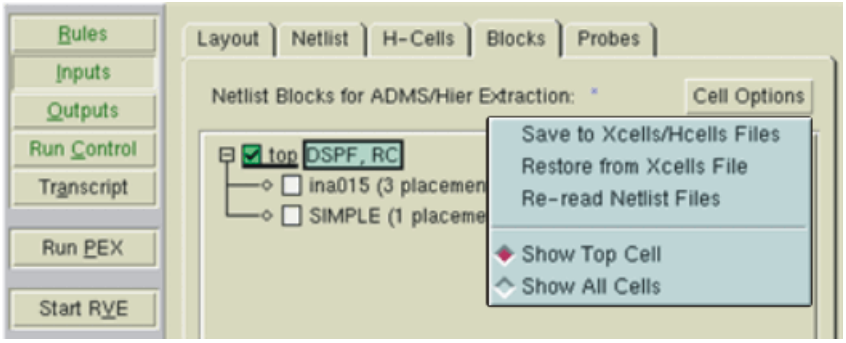
c. To change from the default RC extraction mode to RCC or R, right-click the green box with the formats. This menu also lets you specify the netlist name and mark a block for cross-annotation.

**Figure 6-12. 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.

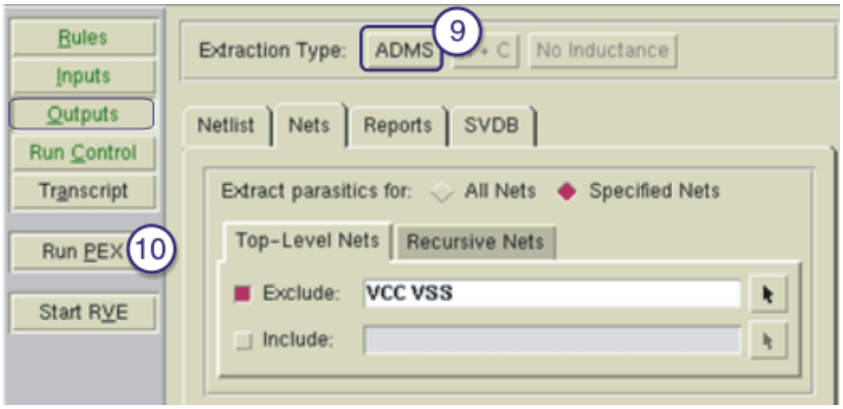
**Figure 6-13. Saving Blocks to Cell Files**




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

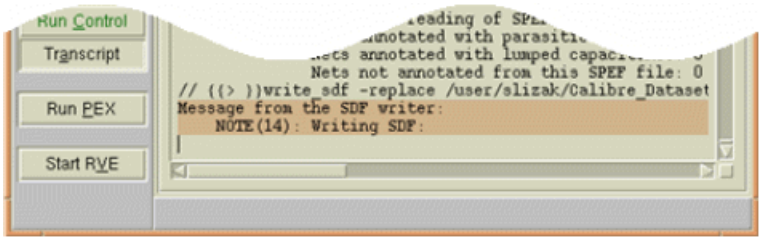
may also want to exclude power and ground nets from extraction. This is set under the **Nets** tab.

**Figure 6-14. 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 6-15. Transcript for Successful ADMS Run**




## Task Validation

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" on page 271

# 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

### Requirements

- 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 Prerequisites for Performing Parasitic Extraction.

### Procedure

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.

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
   ```

### Task Validation

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 only specifies the settings particular to creating a Primetime netlist.

## Requirements

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

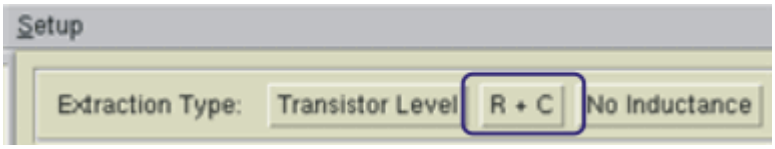
  For more information refer to Prerequisites for Performing Parasitic Extraction.

## Procedure

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

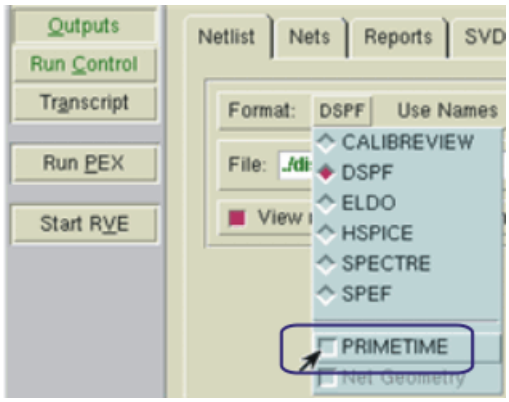
   Click the **Outputs** button in the left pane. In the area above the tabs, set Extraction Type to **R + C**.

**Figure 6-16. R + C Mode Setting**




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

**Figure 6-17. Primetime Output Setting**




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

## Task Validation

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

### Requirements

- 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 Prerequisites for Performing Parasitic Extraction.

### Procedure

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.”

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
   ```

### Task Validation

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" in Output Reference

# Creating a Spectre Netlist from Calibre Interactive

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

## Requirements

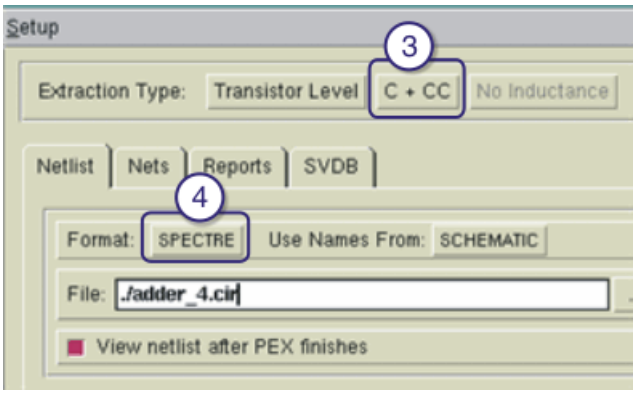
- A valid PEX rule file for this layout.
- Layout database that is LVS-clean.

  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 type. Click the **Outputs** button in the left pane. In the area above the tabs, set Extraction Type to **C + CC**.
4. Select the **Netlist** tab, and set the format to **SPECTRE**.

**Figure 6-18. Extracting a Lumped Capacitance Spectre Netlist**




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

## Task Validation

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

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.

### Requirements

- 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 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 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
   ```

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.

## Task Validation

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" on page 137

# Mixing Parasitics from Calibre Interactive

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

## Requirements

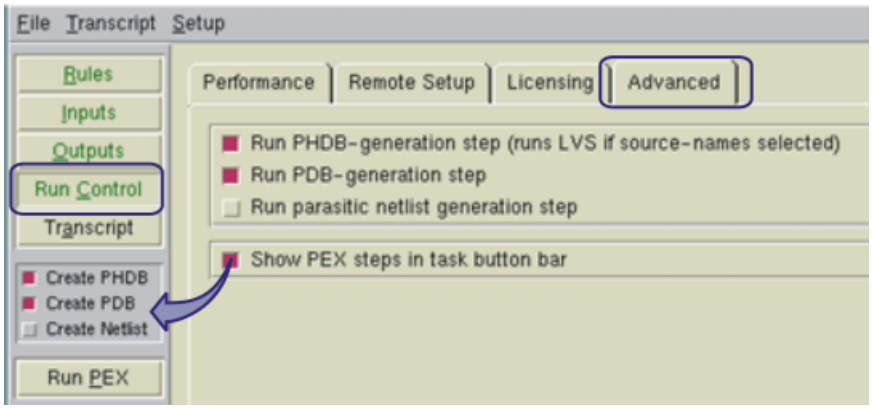
- A valid PEX rule file for this layout.
- Layout database that is LVS-clean.

  For more information refer to Prerequisites for Performing Parasitic Extraction.

## Procedure

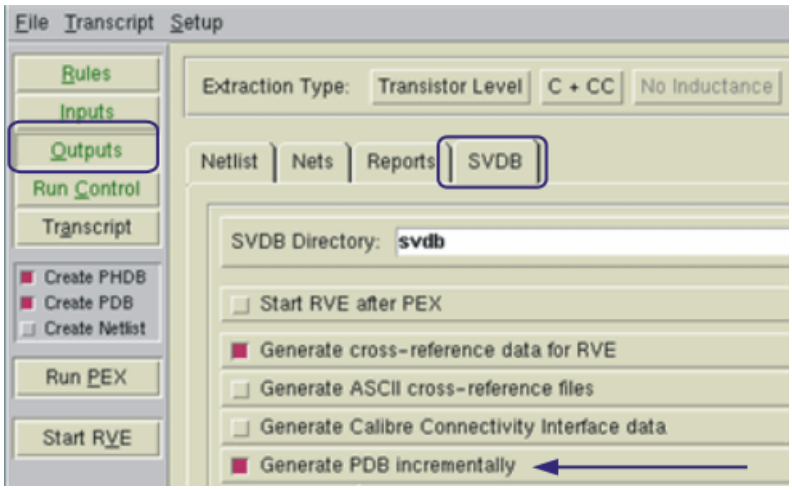
1. Start the PEX interface in Calibre Interactive.
2. Load a runset or rulefile.
3. Enable the PEX step controls.

**Figure 6-19. 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 6-20. Generating PDB Incrementally**



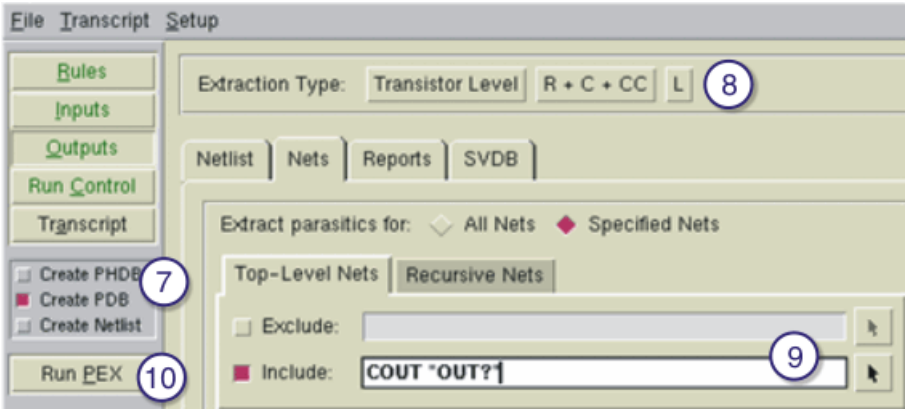

a. Click the **Outputs** button in the left pane.

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 6-21. 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**.

## Task Validation

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 Interactive User's Manual*

*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

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.

### Requirements

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

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

### Task Validation

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.

## Requirements

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

  For more information refer to Prerequisites for Performing Parasitic Extraction.

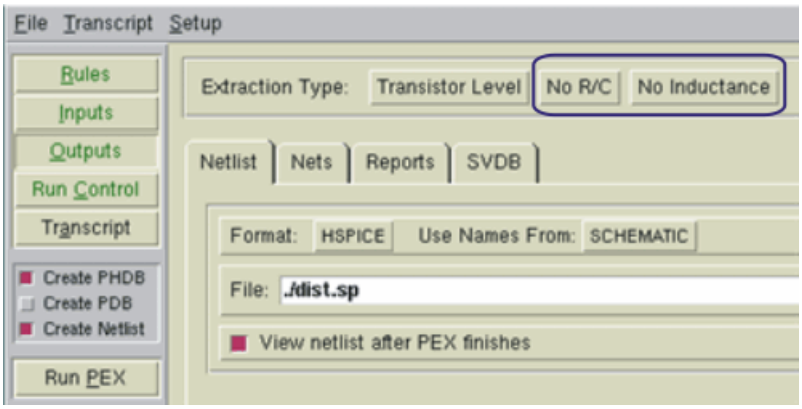
## Procedure

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; this figure happens to show transistor level.)

**Figure 6-22. 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.

## Task Validation

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

### Requirements

- 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_flag rules
   ```

   where ***parasitic_flag*** 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_flag*** 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
   ```

## Task Validation

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.

## Requirements

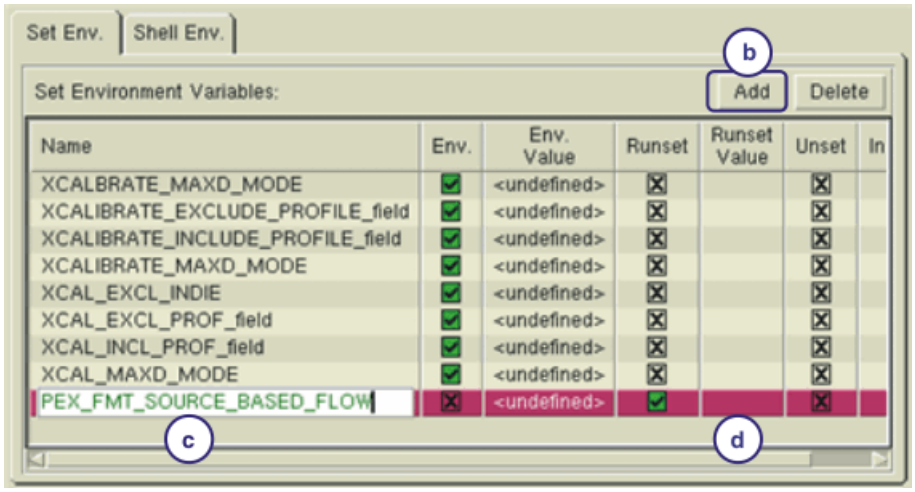
- A valid PEX rule file for this layout.
- Layout database that is LVS-clean.

  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. Set the environment variable PEX_FMT_SOURCE_BASED_FLOW to ON.
   a. From the Calibre Interactive menu bar, select **Setup > Set Environment**. The Set Environment Variables dialog opens.

**Figure 6-23. Setting Environment Variables in Calibre Interactive**




   b. Click the **Add** button to add a new entry.
   c. Type PEX_FMT_SOURCE_BASED_FLOW as shown in the figure.
   d. In the Runset Value column, double-click and type ON.
   e. Click Apply.
   f. Click OK to close the dialog.

4. Specify the extraction type and other settings. Generally, for backannotation you use names from schematic.
5. Click **Run PEX** to produce the netlist.

## Task Validation

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 Interactive User's Manual*

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.

## Requirements

- 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.)

## Task Validation

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

“Examples” in *Standard Verification Rule Format (SVRF) Manual*

# 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

### Requirements

- 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.

### Task Validation

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

"Examples" in *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.

## Requirements

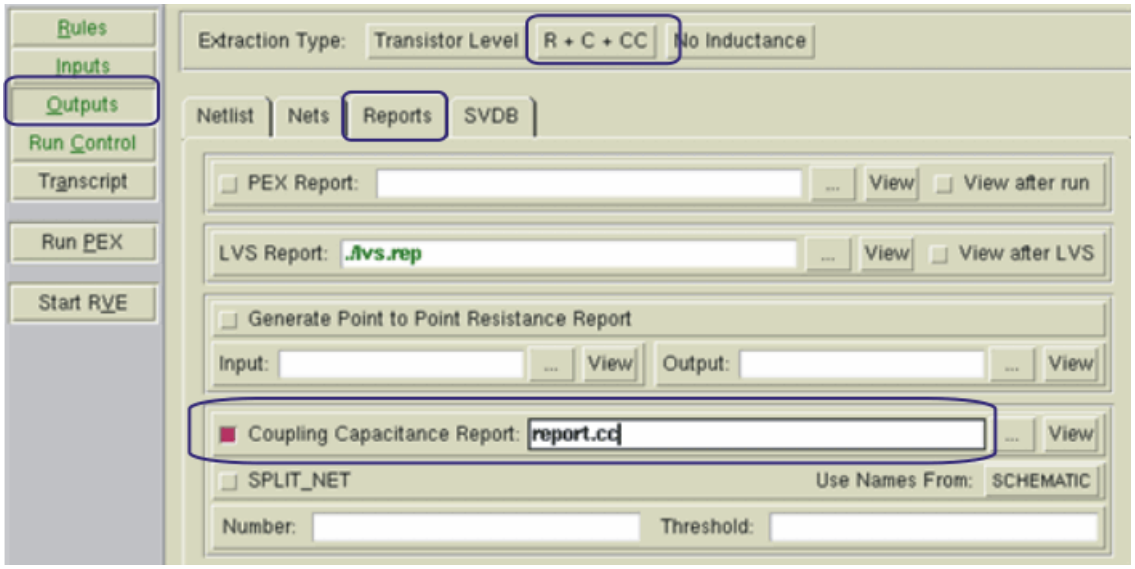
- A valid PEX rule file for this layout.
- Layout database that is LVS-clean.

  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 6-24. 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 not report.

6. Set other controls as needed.
7. Click **Run PEX** to produce the report (and netlist).

## Task Validation

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

"Examples" in *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 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

### Requirements

- 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 *name signature* | Name of the resistor pin |
     | PORT \| PROBE *p_name* | Name of port or probe |
     | HPORT *path name* | The path to hport and name of hport |
     | 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.

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

## Task Validation

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

“Examples” in *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.

## Requirements

- 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

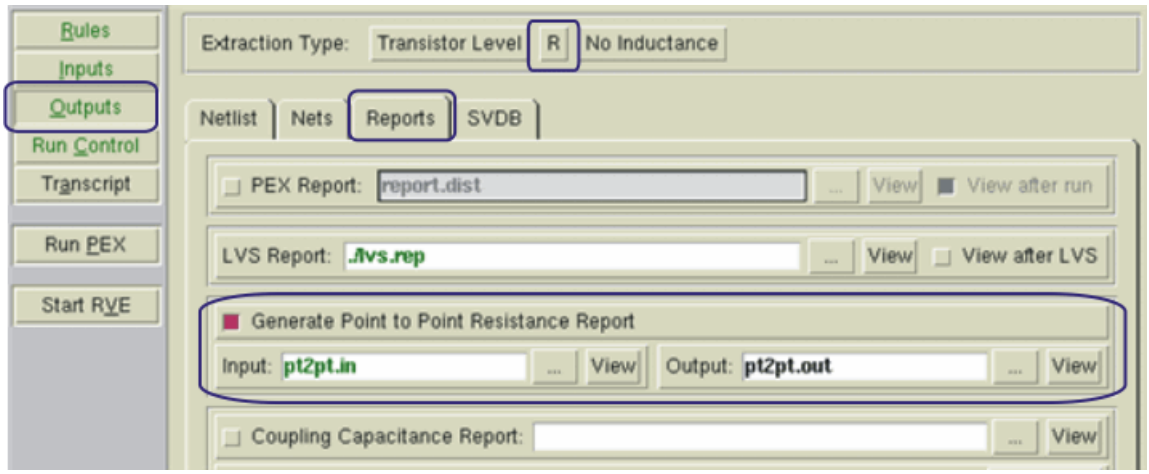
   - *Netname* is the *layout* name of the net, and the same for both entries. (Resistance cannot be measured across devices.)
   - *Location* is one of the following:

     | | |
     |---|---|
     | PIN *name signature* | Name of the resistor pin |
     | PORT \| PROBE *p_name* | Name of port or probe |
     | HPORT *path name* | The path to hport and name of hport |
     | 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 6-25. Point-to-Point Resistance Report Settings**




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

## Task Validation

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

“Examples” in *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.

## Requirements

- 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” on page 123.

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_flag rules
   ```

   where *parasitic_flag* 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
   ```

## Task Validation

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" in Types of 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" on page 62 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

### Requirements

- 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 calibrexrccb license.

**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_flag -cb rules
   ```

3. Generate the netlist.

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

### Task Validation

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 CB” in the *Calibre Verification User’s Manual*

“Calibre Cell/Block” in the *Calibre Administrator’s Guide*

# 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.

## Requirements

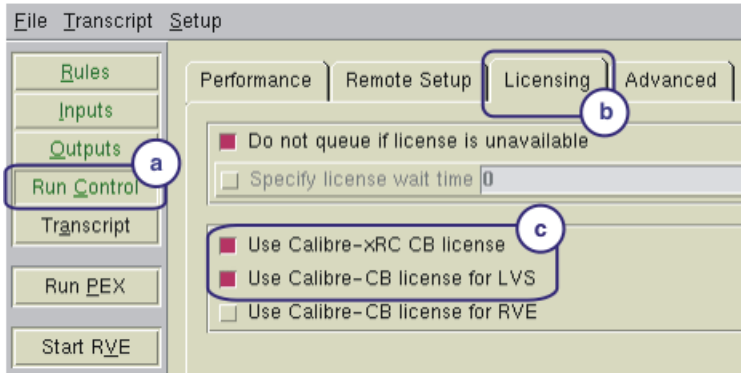
- A valid PEX rule file for this layout.
- A calibrexrccb license.
- Layout database that is LVS-clean.

  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. Enable CB license acquisition.
   a. Click the Run Control button.

**Figure 6-26. 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**.

## Task Validation

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 "Running:" These lines show the command line that was executed and will include a -cb switch when using the CB license.

## Related Topics

"Calibre CB" in the *Calibre Verification User's Manual*

Getting Started: Parasitic Extraction Using Calibre Interactive

# Chapter 7
# Running The Calibre xRC Tool With ASIC Designs

The ASIC features in the Calibre xRC tool are useful for extracting parasitics for digital ASIC designs with millions of nets and devices. These designs are rendered using auto-place-and-route software and contain standard cells connected with Manhattan-style routing.

ASIC mode is optimized to improve performance for extracting parasitics for this type of layout. When compared to flat or non-ASIC hierarchical extraction, using ASIC mode reduces runtime, lowers memory usage, and reduces the parasitic netlist size.

ASIC mode supports LEF/DEF and LEF/DEF combined with GDS data for subcells and metal fill. It also supports Milkyway and GDS-only data.

This chapter discusses specific topics related to parasitic extraction with ASIC designs:

# ASIC Mode Extraction Features

ASIC mode is invoked by using the -asic command line option in the extraction stage (-pdb) when using Milkyway or GDS layout data. It is invoked automatically when using LEF/DEF data.

## In-Die Variation

In-die variation and metal fill extraction are supported. Different combinations of LEF/DEF and GDS, LEF/DEF only, GDS only, and Milkyway data formats can be used together with ASIC mode. DSPF and SPEF output formats which are used for Static Timing Analysis, or STA, are supported.

## Extraction Modes

ASIC mode supports lumped capacitance (-c), distributed resistance and capacitance (-rc), and distributed resistance with coupled and intrinsic capacitance (-rcc) extraction types.

## Excluded And Selected Nets

Excluding user-selected nets during extraction is supported with ASIC mode. Excluded nets are not removed from the layout database. Their effect on signal nets is included in the parasitic netlist. However, the final netlist will not include excluded nets or parasitics attached to these nets. For more information on how to exclude nets from extraction, see PEX Extract Exclude in the Standard Verification Rule Formant (SVRF) Manual.

ASIC mode supports extracting parasitics for a selected net (-select command line option). If this is required and ASIC mode is enabled, Calibre xRC will include the selected net with its associated parasitics in the output netlist.

## Parasitic Netlist Reduction

ASIC mode works with the SVRF reduction statements available for the Calibre xRC tool. Reduction SVRF statements begin with the key words “PEX Reduce”. Useful reduction statements for ASIC mode in order of application include:

- PEX Reduce Digital
- PEX Reduce CC
- PEX Reduce Mincap and PEX Reduce Minres
- PEX Reduce TICER

PEX Reduce Digital is applied during ASIC mode extraction. You can override the default values by specifying this statement in the rule file. Use the other reduction statements if specific

reduction is needed. For more information, see the *Standard Verification Rule Format (SVRF) Manual*.

SVRF reduction statements for a Calibre xRC run are executed during the formatter stage (-fmt). You can adjust reduction parameters then rerun the formatter stage without rerunning the the parasitic extraction stage (-pdb) or connectivity extraction stage (-phdb or -lvs).

## PEX Reduce Digital

PEX Reduce Digital is automatically applied when using LEF/DEF layout data, unless you use the -noasic switch during the parasitic extraction stage (-pdb).

You can control the reduction results by using the DELAY_ERROR and NOISE_ERROR parameters for PEX Reduce Digital. DELAY_ERROR specifies a user-defined timing delay threshold while NOISE_ERROR specifies a user-defined noise amplitude threshold. For more information and examples of PEX Reduce Digital, see the *Standard Verification Rule Format (SVRF) Manual*.

## PEX Reduce CC

For distributed RC network extraction (-rcc), using PEX Reduce CC reduces coupled capacitance between nets based on a ratio of the total net capacitance. For a reasonable reduction you may use:

```
PEX REDUCE CC RATIO 0.3
```

The 0.30 value means that any parasitic coupling capacitor below 30% of the total capacitance on the coupled nets will be grounded.

For more information on PEX Reduce CC, see "Capacitive And Resistive Reduction" in this manual and PEX Reduce CC in the *Standard Verification Rule Format (SVRF) Manual*.

## PEX Reduce Mincap and PEX Reduce Minres

The PEX Reduce Mincap statement can be used for reducing intrinsic and coupling capacitance during the formatter stage (-fmt). A useful invocation of the statement may be:

```
PEX Reduce Mincap COMBINE 1
```

Assuming that the unit capacitance for the design is femtofarads, this statement sets a threshold of 1 femtofarad for combining intrinsic and coupling capacitances on nets. Any parasitic capacitors below this threshold will be combined with neighboring parasitic capacitors.

The PEX Reduce Minres COMBINE statement can be used for combining parasitic resistors based on a user defined threshold. A useful invocation of this statement may be:

```
PEX Reduce Minres COMBINE 0.1
```

Assuming that the unit resistance for the design is in ohms, this statement sets a threshold of 0.1 ohm for combining parasitic resistors on the same net.

For more information on PEX Reduce Mincap and PEX Reduce Minres, see "Capacitive And Resistive Reduction" and the *Standard Verification Rule Format (SVRF) Manual*.

### PEX Reduce TICER

PEX Reduce TICER provides high accuracy reduction in the Calibre xRC tool. PEX Reduce TICER may not reduce the parasitics below an internal software limit which is set by the RC network topology and internal software parameters. The effect of these limits is that you may not see a reduction in netlist size regardless of how you set the frequency parameter. For more information on PEX Reduce TICER, see "TICER" in this manual and PEX Reduce TICER in the Standard Verification Rule Format (SVRF) Manual.

## Black Box And Gray Box Extraction

The PEX Xcell Extract Mode SVRF statement specifies how geometries inside cells in the xcell file are handled during extraction.

Black box extraction mean that the internal contents of the xcells are ignored during the parasitic extraction of interconnect. Gray box extraction means that the parasitics between the top level routing and cell geometries are not ignored. These parasitics are added to the intrinsic capacitance of a net crossing over the cell. When using ASIC mode, gray box mode is used by default.

You can also control how routing obstructions inside standard cells are handled during parasitic extraction. For more information on how to account for geometries inside standard cells, see the PEX Xcell Extract Mode and PEX DEF Extract Cell Obstructions SVRF statements.

## Controlling Extraction Hierarchy In ASIC Mode

In ASIC mode, hcell and xcell files can be used to control extraction with LEF/DEF, GDS, and Milkyway data in different ways. Although the content of an xcell file is similar to the content of a hcell file, the xcell file may also contain specific flags used only for parasitic extraction. For more information on how to set up an xcell file, see "Controlling Hierarchy with Xcells".

### Hierarchy In LEF/DEF

When using LEF/DEF data with ASIC mode, no hcell or xcell file is required because the Calibre xRC tool automatically identifies the design hierarchy. The LEF data is used to generate the hcell mapping information internally. This generated list of cells is black boxed during extraction.

If you need to append information to the hcell list automatically generated from the LEF/DEF data, use the -hcell command line option to add more cells to the hierarchy list during the PHDB stage. You cannot remove cells from the hierarchy.

If you need to override the internal hcell definitions, use -hcell during -lvs or -phdb step, or -xcell during the PDB stage.

### Hierarchy In GDS And Milkyway

If GDS or Milkyway data is used with ASIC mode, a hcell file must be specified during connectivity extraction (-phdb or -lvs) and an xcell file must be specified during parasitic extraction (-pdb).

When using these data formats, the Calibre xRC tool by default uses devices defined in the LVS rules as the lowest level of hierarchy in the design. To take advantage of ASIC mode, you must define the hierarchy of the design during the connectivity extraction and parasitic extraction stages. These files should contain all standard cell and custom cell mapping between layout and source.

If using the -xcell option, the xcell file must contain the same cell mapping as the hcell data so that extraction is run only on the interconnect.

## Extraction for Static Timing Analysis (STA)

ASIC mode supports extraction of SPEF and DSPF netlists which are used in Static Timing Analysis (STA) tools such as PrimeTime. For example, to extract a SPEF netlist, include the following SVRF statement in your extraction rule file:

```
PEX NETLIST SPEF ‘asic.spef’ PRIMETIME SOURCENAMES
```

where the PRIMETIME parameter specifies that the SPEF output meets Primetime requirements and SOURCENAMES specifies to use the source netlist names in the output.

For more information about controlling the output netlist, see PEX Netlist in the Standard Verification Rule Format (SVRF) Manual.

## ASIC Mode and Calibre Interactive

ASIC mode works in conjunction with the Calibre Interactive RVE viewer to display results. The results can be reviewed in layout viewers such as Mentor Graphics IC Studio, Calibre DESIGNrev viewing environment, or other third party viewers such as Cadence Virtuoso Layout Editor.

# LEF/DEF Data In ASIC Mode

The ASIC mode is automatically selected when using LEF/DEF data. If you do not need to use this mode with LEF/DEF, then you must specify the -noasic switch during the PDB stage.

When using LEF/DEF data, you do not have to perform LVS before extracting parasitics. The DEF format contains the source device names, net names, and the physical routing, making layout netlist extraction and comparison (LVS) unnecessary. You must perform the PHDB generation stage to read the LEF/DEF data into the Calibre database. Hcell information is automatically generated during the PHDB stage. If you need to override this, then specify a hcell file during the PHDB and an xcell file during the PDB steps.

ASIC mode supports designs that use hierarchical DEF. The top-level cell can reference LEF library standard cells, user-defined LEF abstract cells, or separate DEF files that contain LEF cells along with place-and-route data. Calibre xRC is able to assemble the top-level netlist from a design that is partitioned into sub-blocks by using hierarchical DEF. Both the DEF and LEF sub-block data and LEF standard cell data are required. The LEF abstract and DEF file must be present in the design library and must be included in the hcell information.

For more information on how to use LEF/DEF input, see "Using LEF/DEF Input" on page 125.

## LEF/DEF With GDS Cell Data

The ASIC mode supports extracting parasitics with a mixture of LEF/DEF and GDS data. You may want to do this if your design includes large circuits, such as memory or analog blocks, and need to extract a parasitic model that represents detail from the geometries inside such blocks. You can do this by including a GDS file in the extraction. The GDS view of a block must have the same number of pins as the LEF view. You will need to use gray box extraction mode when including GDS data with LEF/DEF.

Both LEF and GDS data for the block of interest are required. The LEF abstract must be present in the LEF files for the design and the GDS file for the block will need to be included using a hcell file. For example, if your design includes a GDS file for a RAM block, the xcell file would include the following entry:

```
64kRAM -gds layout_path/64kRAM.gds
```

The block 64kRAM has both LEF and GDS data. This xcell declaration specifies that the file 64kRAM.gds, found in layout_path, will be used instead of the 64kRAM LEF data.

## LEF/DEF With DEF Metal Fill

ASIC mode supports DEF format FILLS statement in LEF/DEF data for including metal fill in the design. FILLS generates floating metal fill which is read during the PHDB stage of extraction. You can then use the PEX Extract Floating Nets SVRF statement to customize how the metal fill is handled during extraction.

## LEF/DEF With GDS Metal Fill

ASIC mode supports the combination of LEF/DEF data and GDS data for metal fill. Normally, using LEF/DEF data in extraction does not require a hcell or an xcell file because the Calibre xRC tool automatically recognizes the design hierarchy. However, when including GDS data for metal fill, you will need to use an xcell file which specifies the GDS file and file location. This xcell file will contain this single entry.

This example shows how the GDS file is included in an xcell file:

```
DSPTop -fill layout/gds/metal_fill.gds DSPTop
```

where DSPTop is the LEF/DEF top-level design, metal_fill.gds is the GDS file found in the layout/gds path, and DSPTop is the top cell name in the GDS file.

## Shorted Routes In LEF/DEF

ASIC mode supports parasitic extraction of shorted nets in LEF/DEF data. In the DEF format logical and physical connectivity are separate. This makes it possible for the router to introduce logically correct but physically incorrect connections during the routing stage. For example, you may have two metal one lines crossing each other when connecting pins of different cells. These are separate logic signals, but physically shorted together.

With LEF/DEF data, the Calibre xRC tool will retain the logical connectivity during extraction and will not represent the short in the output netlist. The short may slightly affect the extracted intrinsic and coupled capacitances as well as the extracted resistance. However, only the parasitics in the immediate area of the short will be affected. These effects are dependent physical characteristics of the shorted nets.

With GDS and Milkyway data, the output netlist is affected. Shorted nets alter the logical connections between devices and this is reflected in the output netlist.

# Milkyway Data In ASIC Mode

Using Milkyway layout data with ASIC mode requires the -asic switch during the PDB stage:

```
calibre -xrc -pdb -asic ...
```

The Calibre xRC tool accepts Milkyway database layouts through the Foreign Database Interface (FDI). There are two possible ways to read a Milkyway database:

- Direct Read Flow
- FDI GDS Flow

## Direct-Read Flow

This method uses Calibre FDI to generate the PHDB for the design by directly reading the Milkyway design database. The layout database type and a layer mapping file must be specified in your SVRF rule file.

The SVRF rule file should contain the following statements:

```
LAYOUT SYSTEM MILKYWAY
LAYOUT PATH mw_db
LAYOUT TOP top_cell
```

where mw_db is the path to the Milkyway database and top_cell is the top level cell name of the design.

The layer mapping file is defined by an environment variable:

```
setenv PEX_FDI_MAP fdi_map_file
```

where fdi_map_file specifies the layer mapping file pathname and filename. This file defines the layer mapping as follows:

```
#FDI_LAYER_MAP    Milkyway_Layer    SVRF_Layer
FDI_LAYER_MAP  met1     metal1
FDI_LAYER_MAP  v1       via1
FDI_LAYER_MAP  met2     meta2
```

For more information, see the PEX_FDI_MAP environment variable.

## FDI GDS Flow

The FDI GDS flow is useful for debugging design database related problems such as missing layers, text layer mapping, and so on. At times, it is helpful during the debugging process to see the layout in the GDS format to ensure database integrity.

The Calibre FDI utility, fdi2gds, can be used to generate a GDS file from the Milkyway database for extraction. The net names are taken from the Milkyway layout database because a source netlist is not used. The net names can be added as GDS text layers or as GDS properties on polygons.

Milkyway database allows ‘/’ (forward slash) to be used in net names. However, this is not supported in the FDI GDS flow. To rename this character, include the following SVRF statements in your rule file:

```
LAYOUT RENAME TEXT ‘#/#.g’
PEX NETLIST CHARACTER MAP ‘./’
```

For more information on Layout Rename Text and PEX Netlist Character Map, see the *Standard Verification Rule Format (SVRF) Manual*.

This is one way to use the Calibre FDI utility from the command line to create a GDS file for your design:

```
fdi2gds -system MILKYWAY -design mw_lib mychip \
-outFile mychip.gds -layerMap mylayermap -logFile fdi2gds.log \
-annotateNets PROPERTY 16 -instPinAsShape -viewList FRAM CEL
```

where

- ***fdi2gds*** invokes the Calibre FDI utility for creating a GDS file.
- ***-system*** is a required switch that specifies the layout system, in this case Milkyway.
- ***-design*** is a required switch that specifies the design library location and top level cell name.
- ***-outFile*** is a required switch that specifies the GDS output file and location.
- ***-layerMap*** is an optional switch that specifies the pathname and filename of a layer mapping file. This is an example of a file that maps Milkyway layers to GDS layers:

  ```
  #input_layer_name output_layer_number datatype
  met1     12    0
  via      13    0
  met2     14    0
  via2     15    0
  met3     16    0
  ```

  In this example, met1 is the Milkyway layer that maps to GDS layer 12 and datatype 0.
- ***-logFile*** is an optional switch that specifies the pathname and filename for the runtime log for translation.
- ***-annotateNets*** is an optional switch that specifies adding annotation to all the shapes that belong to a net. In this example, the annotation is added as a property to layer 16. In the SVRF rule file for extraction using this GDS file, you will need to add the following statement:

  ```
  LAYOUT PROPERTY TEXT 16
  ```
- ***-instPinAsShape*** is an optional switch that specifies allowing pins as shapes in the parent cell.
- ***-viewList*** is an optional switch that specifies the order for opening views for lower-level cells.

For more information on using the fdi2gds utility, see "Translate Third-Party Layout Databases" in the *Calibre Verification User's Manual*.

## Case Sensitivity In Milkyway Data

Milkyway database format supports case sensitivity, so you may have to include the following SVRF statements in your rule file:

```
LAYOUT CASE YES
LAYOUT PRESERVE NET CASE YES
```

# GDS Data In ASIC Mode

Using GDS layout data with ASIC mode requires the -asic switch during the PDB stage. If the GDS design data does not pass the heuristics requirements for ASIC mode, then Calibre xRC uses normal extraction (-noasic). Using GDS data with ASIC mode automatically invokes gray box extraction.

If you want to use the Calibre xRC ASIC mode with GDS data, you must run the LVS step with a hcell file that contains the standard cells and custom blocks in the design. This step is necessary for creating the instance and name cross reference files which the formatter uses to generate a parasitic netlist based on the source. An exception to this is when you are using annotated GDS data (AGDS) which contains the connectivity information. With AGDS data, you can use the -phdb command line switch.

If the GDS file contains standard cell abstracts, then the cells need to be specified in both the hcell file and the LVS Box SVRF statement during LVS. The LVS step will report that the design matches the source netlist, even though connectivity information from the standard cells is not extracted. Use LVS Box if the schematic or layout for a cell is not complete or only the schematic symbol or layout outline with pin locations is available.

Using GDS data with ASIC mode always requires a hcell file. If the GDS file contains transistor-level data for standard cells, then a hcell file containing all of the standard cells is required. If the GDS file or the source netlist does not contain full transistor-level data for all of the standard cells, then you can use the LVS Box SVRF statement during LVS comparison.

# Chapter 8
# Handling Input

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



## Controlling Hierarchy with Xcells

When performing hierarchical extraction, you define your design's hierarchy for the Calibre xRC tool 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.

### The Xcell List

An xcell list is an ASCII file you create[1]. 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, the Calibre xRC tool 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, the Calibre xRC tool 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***

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.

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

## Calibre nmLVS-H and Calibre xRC Cell List Compatibility

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

```
layout_name source_name
```

With the Calibre nmLVS-H tool, 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 tool run using the same cell list throughout, called *xcell_list*:

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

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

- Calibre nmLVS-H tool — maps the layout's cell names to their corresponding source's cell names when you perform source name extraction. When invoking the Calibre nmLVS-H tool, you specify the hcell list by using the "-hcell *xcell_list*" switch. See "calibre -lvs" in Appendix 13, "Calibre xRC Tool Invocation Reference." See "Hcells" in the *Calibre Verification User's Manual*.
- Calibre xRC tool — defines a design's hierarchy for global net extraction. You create an xcell list based on the hierarchy you want, and the Calibre xRC tool analyzes and extracts the hierarchy.

Although an hcell list can be used as an xcell list, xcell lists have additional options. Xcell entries can use wildcards to match multiple cells. Entries can also specify specific cell instances to extract "in context".

## Creating an Xcell List

An xcell list can have either of two formats. 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 |
                           -P [-GDS gds_macro] |
                           -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 will 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 8-1 provides more information on each flag.

**Table 8-1. Xcell flags**

| Flag | Usage |
|---|---|
| -C *layout_path* | Used for in-context extraction as explained in "In-Context Extraction" on page 44. If doing gate-level extraction, the -C is ignored. See "Discovering Layout Paths for In-Context Cells" to determine a valid *layout_path*. |
| -P | Indicates the cell is a primitive. The contents are not extracted. In gate-level extraction, all cells are treated as primitives. |
| -GDS *gds_macro* | Used only with LEF/DEF designs. Indicates that a cell is a GDS macro. Can only be specified in conjunction with -P. |
| -FILL *file.gds cellname* | Used only with the top cell of a LEF/DEF design. Specifies that metal fill is in a separate file, *file.gds*, in cell *cellname*. |
| -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    //use gds layout for extraction
INV      INV            //handling depends on the extraction type
```

**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 list.

## Discovering Layout Paths for In-Context Cells

To specify a valid layout path or "parent_id/cell_id" for -C cells in the xcell file:

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 Calibre® 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.

**Example 8-1. Wildcards in an Xcell List**

| Xcell file without wildcards | Same Xcell file with wildcards |
|---|---|
| pmos_rf1<br>pmos_rf2<br>pmos_rf3<br>pmos_rf7<br>...<br>nmos_rf1<br>nmos_rf2<br>nmos_rf3<br>nmos_rf5<br>... | pmos_rf*<br>nmos_rf* |

## 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 set.

# Using LEF/DEF Input

To run the Calibre xRC tool with LEF/DEF designs, you need to specify specific SVRF statements and set up mappings before you invoke the software.

1. Set Up the SVRF Statements.
2. Resolve Different Layer Naming Conventions.
3. If the layout references GDS blocks, List GDS Files.
4. Run from the Command Line.

# Set Up the SVRF Statements

To implement the LEF/DEF flow, add the following Layout statements to your rulefile:

```
LAYOUT PATH "lef_input" "def_input"
LAYOUT SYSTEM LEFDEF
LAYOUT PRIMARY "PRIMARY CELL"
LAYOUT USE DATABASE PRECISION YES
```

The Layout Path statement can handle multiple LEF or DEF files. The files can either be specified individually or by directory. If you use directories, supply the directory names in the Layout Path statement instead of specific file names. There can be only one LEF directory and one DEF directory per run.

In the LEF file directory, the files should end in ".lef" or ".tlef". For the DEF file directory, all files should end in ".def".

In the LEF/DEF flow, cross references are automatically generated, so in all SVRF statements requiring a source/layout designation, always specify *source*. For example:

```
PEX NETLIST filename DSPF SOURCE LOCATION
```

If the design includes GDS blocks, specify case-sensitive naming. Add the following statements to your rulefile:

```
LAYOUT CASE YES
SOURCE CASE YES
```

# Resolve Different Layer Naming Conventions

If the layers in the LEF/DEF layout do not match the layer names in the Connect statements, the DEF layers are discarded with a warning. To correct this, use a mapfile to map different names in the LEF/DEF files to the SVRF rule file.

You can use the Layer SVRF statement to make this correction. See "LEF/DEF Layer Mapping" subtopic in the Layer SVRF statement description.

# List GDS Files

If you are using a LEF/DEF design that has GDS blocks or uses GDS files for fill, you need to create a list which will be merged with the hcell and xcell files when you create the PHDB.

## Metal Fill

Metal fill is specified as part of the top DEF cell and can only appear once in the list. The entry should have the following format:

```
def_cell      -fill gds_file cell_containing_metal_fill
```

### GDS Macros

The entries should have the following format:

```
macro_name    -gds gds_file    -p
```

where *macro_name* is the macro of the LEF file and *gds_file* is the name of the GDS file containing the associated cells. The macro name can use "*" to match any text. If all GDS blocks are in a single file, your list can consist of the single entry:

```
*             -gds gds_file    -p
```

The GDS filename cannot use wildcards. The ports of the GDS cell should match the macro; port names are case sensitive. Macro names must appear only once.

## Run from the Command Line

After you have set up your SVRF rule file, mapped layers, and listed any GDS files in *gds_list*, you extract and netlist using the layout-based flow. You need to use the layout-based flow instead of the source-based flow because the source statements do not support LEF/DEF.

1. Create the PHDB using layout names and the GDS list as follows:

   ```
   calibre -xrc -phdb [-hcell gds_list] rules
   ```

   For mixed GDS/LEF designs, use the -hcell flag to include the GDS file list. This step creates an xcell list named *top*.xcell in the pex.db directory of your SVDB directory.

2. Perform extraction and create the PDB. For LEF/DEF input, only gate-level extraction is supported. The xcell information is provided automatically.

   ```
   calibre -xrc -pdb [-xcell SVDB/pex.db/top.xcell] [-noasic] -rc rules
   ```

   For mixed GDS/LEF designs, use -noasic to turn off ASIC optimizations. The ASIC optimizations assume regular cell placement and are used by default on LEF/DEF layouts.

   The parasitic switch, -rc, is supplied as an example; you can extract any type of parasitic.

3. Produce the netlist:

   ```
   calibre -xrc -fmt -all rules_file
   ```

Results are written to the file in the format you specified in the PEX Netlist statement.

# Modeling Slotted Metal

By default, the Calibre xRC tool does not differentiate slotted metal from non-solid polygons. You can control how slotted metal is analyzed with the PEX Slots Handling SVRF statement.

Turning on slotted metal modeling causes extraction to complete more quickly by “unslotting” the metal regions. This replaces a polygon with slots, or holes, with a resistively equivalent solid polygon for calculations. The SVRF statement needs to be included in the extraction rules file.

When a polygon on a resistive layer is recognized as slotted, its holes are removed and the Calibre xRC tool calculates the resistance of the newly unslotted polygon, using a higher sheet resistance based on the percentage of hole area to total area of the polygon. The unslotted polygon is also used for capacitance extraction. In most cases, this has minimal effect on accuracy.

For example, for polygons in the metal1 layer to be treated as slotted, specify PEX Slots Handling as follows:

```
PEX SLOTS HANDLING METAL1 COUNT_THRESHOLD 15 AREA_RATIO 0.5
```

A COUNT_THRESHOLD of 15 sets the minimum number of discrete holes required for a metal1 polygon to be treated as slotted. An AREA_RATIO of 0.5 acts as an upper bound; if the area of the holes represents more than that percentage of the total area, the metal is *not* considered slotted. In this case, the upper threshold is 50%.

Figure 8-1 shows an example of a slotted polygon which meets the parameters set by the PEX Slots Handling example. There are 15 slots and the total area of these is not greater than 50% of the polygon area.

**Figure 8-1. Metal1 Polygon Which Meets The PEX Slots Handling Example Parameters**

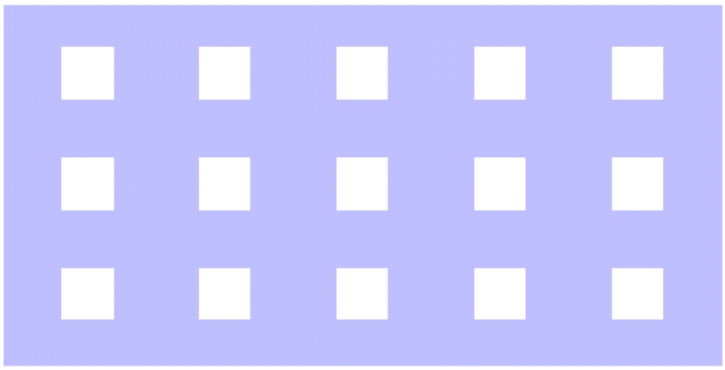


Figure 8-2 shows an example of a polygon that does not meet the parameters of the example. In this case, there are too few slots. This polygon is treated as a non-solid polygon during extraction.

**Figure 8-2. Metal1 Polygon That Does Not Have Enough Slots**

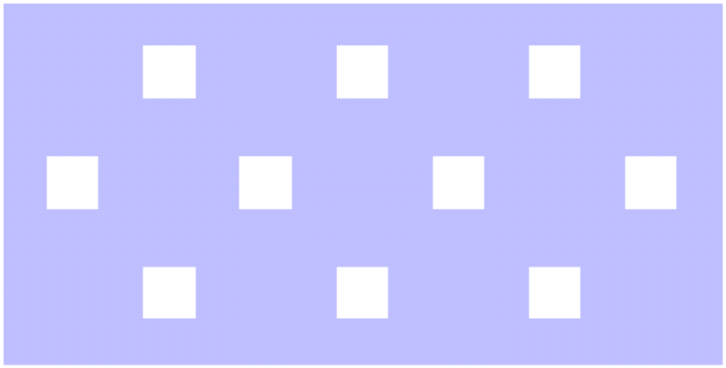
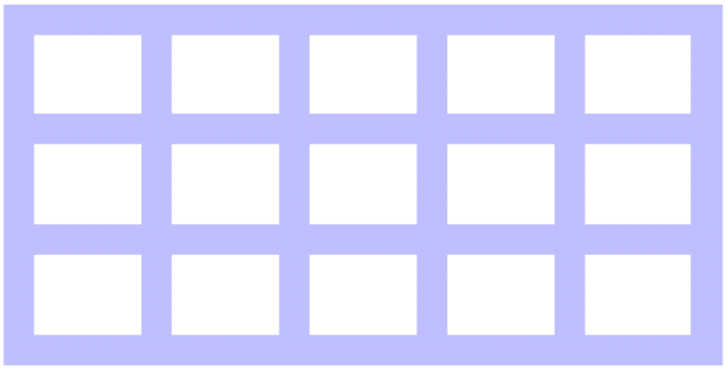


Figure 8-3 shows another example of a polygon that does not meet the parameters from the SVRF statement. Although there are 15 slots, the total area of these is greater than AREA_RATIO of 0.5. The polygon is handled as a non-solid polygon during extraction.

**Figure 8-3. Metal1 Polygon That Has More Than 50% Area in Slots**



# Modeling Metal Fill

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 SVRF 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 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 8-4 shows an example of simple square or rectangular fill between nets drawn on the same metal layer. Figure 8-5 shows multi-layer fill and nets. Figure 8-6 also shows a non-square fill with multiple nets.

**Figure 8-4. Simple Metal Fill On A Single Layer**

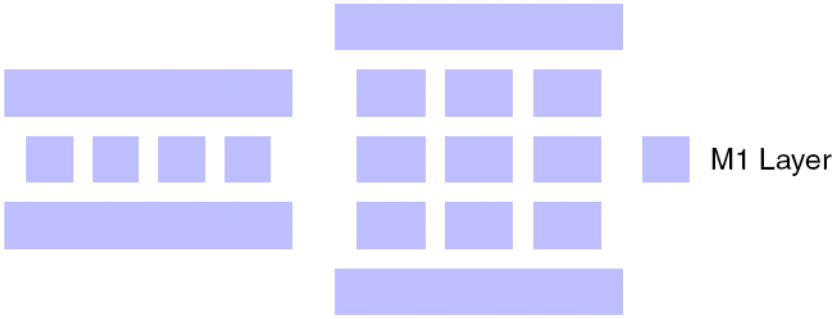


**Figure 8-5. Metal Fill On Multiple Layers with Multiple Nets**

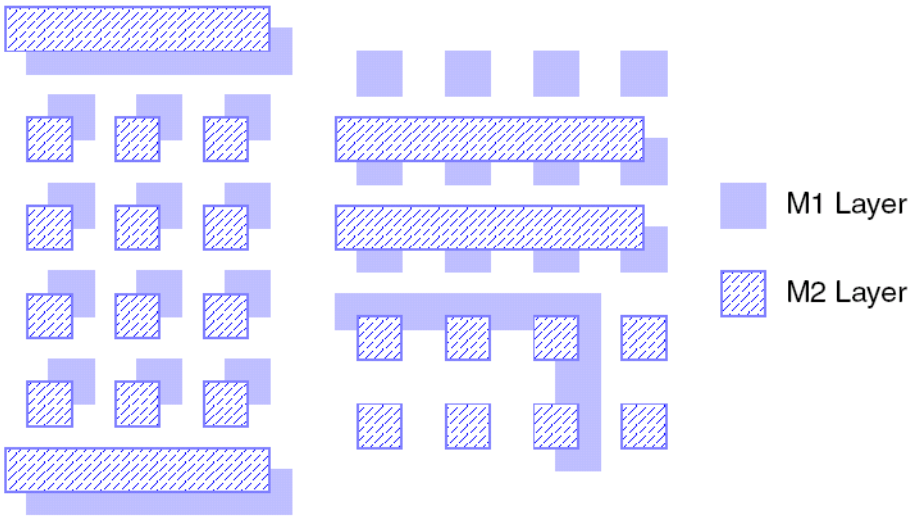
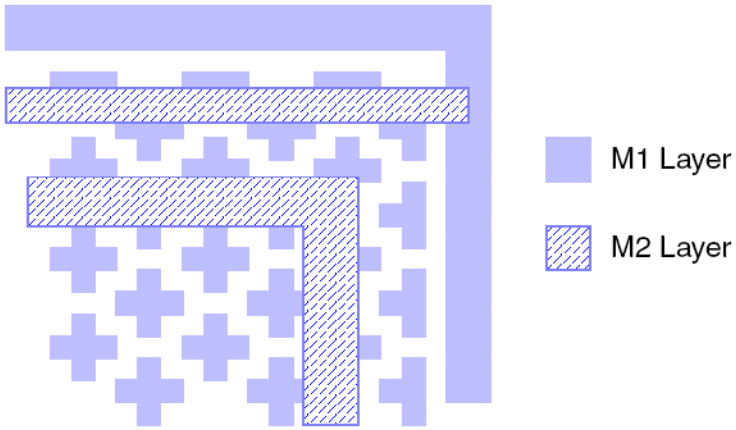



**Figure 8-6. Non-Square Fill With Multiple Nets**




For a more general discussion of floating nets, see “Ignoring or Extracting Floating Nets” on page 139.

# 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 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 SVRF 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 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.

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

## Troubleshooting

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 9
# Tuning Extraction

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



## Extracting Devices without Parasitics

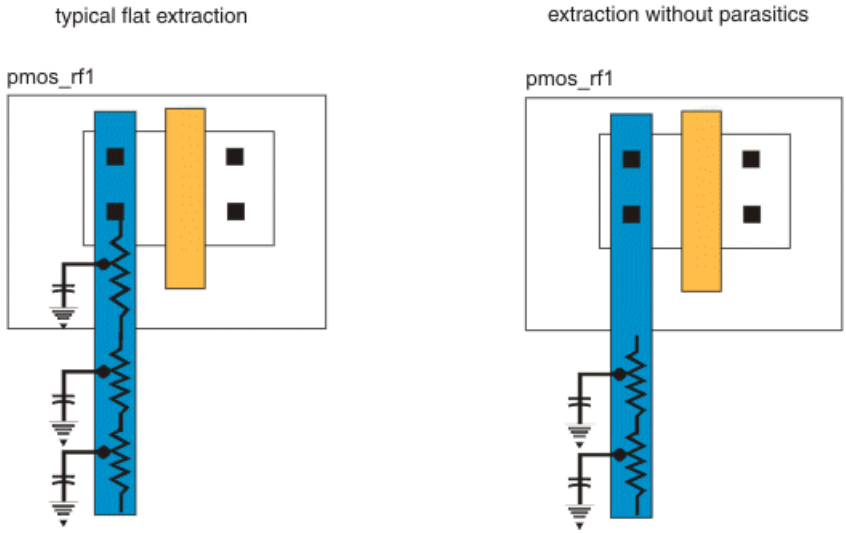
The Calibre xRC tool supports extraction of devices that do not contain parasitics in the xcells. This is useful for flows where the device models already include the parasitics. For example, when physical layout of devices is done by using device generators or parameterized cells (pcells), and the device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

It is not the same as primitive xcells, because the netlist contains the nets for the xcell contents, which have been flattened within the xcell. All nets within the xcell will be ideal nets, that is, they will have no parasitic net models.

Figure 9-1 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 9-1. Device Extraction With and Without Parasitics**




## Invocation

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. (This replaces the PEX Ideal Xcell statement as of version 2008.3.)

## 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.

## 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, since pcells translated to GDSII have random numbers generated after the cell name.

**Example 9-1. Wildcards in an Xcell List**

| Xcell file without wildcards | Same Xcell file with wildcards |
|---|---|
| pmos_rf1<br>pmos_rf2<br>pmos_rf3<br>...<br>nmos_rf1<br>nmos_rf2<br>nmos_rf3<br>... | pmos_rf*<br>nmos_rf* |

# Extracting Particular Nets

When performing selected-net extraction with the Calibre xRC tool, you must perform the following operations:

1. Identify the selected nets by name using the PEX Extract Include SVRF statement in your SVRF rule file.
2. During parasitic database (PDB) creation, use the "-select" Calibre xRC command line switch. (For lumped extraction only, use -cselect to individually report coupled capacitance.)

If you omit the PEX Extract Include SVRF 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.

## Using Wildcards and Specifying Search Level

You can use wildcards to select nets to include or 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 9-2. Including Matching Names in Top Level**

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

**Explanation:** 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 9-3. Including Matching Names From All Levels

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

**Explanation:** 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

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

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 9-4. Excluding Matching Names in Top Level

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

**Explanation:** 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 9-5. Excluding Matching Names From All Levels

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

**Explanation:** 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 invocation. For example:

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

When you use this switch 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:

- Using Floating Net Coupling Algorithm
- Extracting Floating Nets

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

## Using 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 9-2.

**Figure 9-2. Floating-Net Coupling Floating Example**


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 also set). In contrast to extracting floating nets, the final netlist will not contain any floating nets.

**Note**

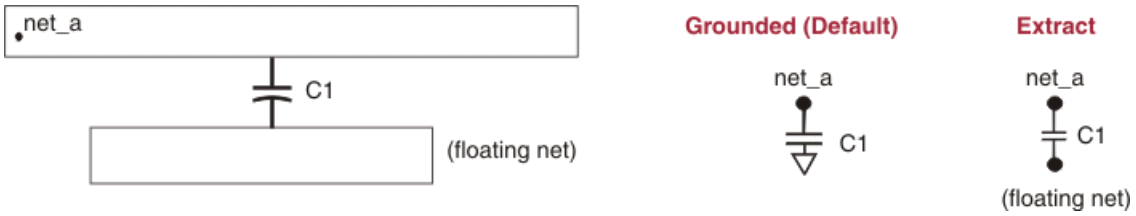
When using selected nets (-select | -cselect switch), nets not selected are assumed to be grounded—this means that only selected nets can potentially float. To use the floating net coupling algorithm when running with selected nets, the floating nets must also be selected.

## 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 9-3 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.

**Figure 9-3. Floating-Net Coupling Extraction Example**




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.

# Chapter 10
# Controlling Netlisting

This chapter describes how to modify the basic extraction procedures to change the produced netlists in the following ways:

### Different Types of Output



### Fixes for Common Issues



### Minor Changes



## Netlisting Multiple Process Corners

Follow this procedure when you want to netlist multiple process 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 process corners, add a PEX Corner Custom statement to your SVRF file.

2. Run the PHDB and PDB steps as normal. All 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,rc_best -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.

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.

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.
3. Run netlisting as usual.

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.

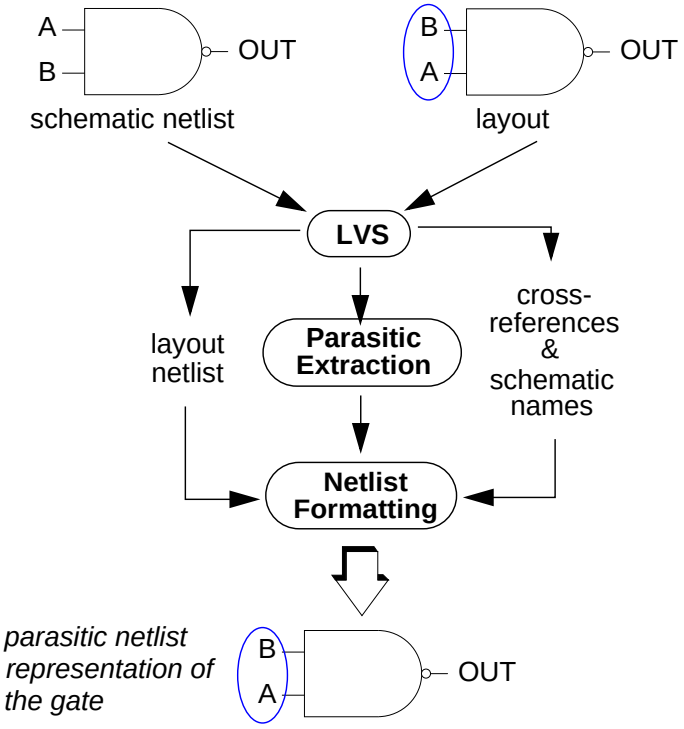
# Correcting Pin Swapping

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 will perform logical pin swapping and propagate 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 10-1 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 10-1. Gate-Level Logical Pin Swapping Example**




In this example, the Calibre nmLVS-H tool will report the design is LVS clean and create the layout-to-source cross-reference files using the swapped pins. By default, the Calibre xRC tool will subsequently use 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” on page 144
- If the pins are on intentional models: “PEX BA Mapfile” in the *SVRF Manual*
- If you are using a layout-based flow and the pins are on primitives or the top circuit only: “PEX Pin Order” in the *SVRF Manual*
- If you need to specify pin direction or pin order on intermediate circuits: “Using Templates” on page 145

# Using the Source Based Flow

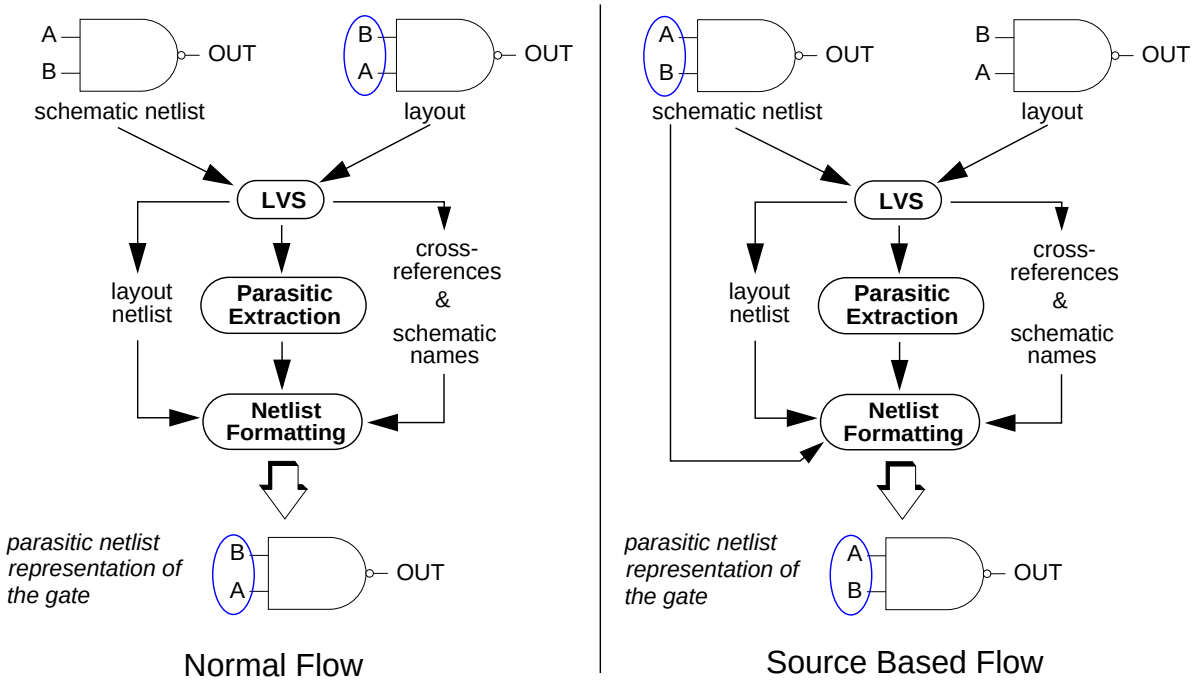
Normally, the Calibre xRC formatter produces a *layout netlist* containing source names. You can correct logical pin swapping in the parasitic netlist using the Calibre xRC formatter's source based flow. When you use this flow, the Calibre xRC formatter produces a *source netlist* containing the extracted parasitics. Source-based flow is not supported for *full hierarchical extraction*.

**Note**

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 10-2 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 Calibre xRC formatter also uses the schematic (source) netlist.

**Figure 10-2. Comparison of Normal and Source Based Flows**




## Modifying the SVRF Rule File

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 in your rule file:

- Mask SVDB Directory BY GATE
- Source Case YES

**Note**

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

## Using Templates

Use templates for defining the pin direction for all subcircuits in layout-based flat and gate-level netlists. In full hierarchical mode, the template only modifies the top level cell. You can define your own templates, or let the Calibre xRC formatter build them and then edit. The syntax is defined in "Templates" on page 264.

**Caution**

The template variable PEX_FMT_HP_PORT_MAP_MODE overrides the PEX Pin Order SVRF statement. If you use templates, you must specify a template for each xcell.

In transistor-level extraction, the template is always created but only used when the variable is set.

### Procedure

1. In a terminal window, set the template environment variable:

   ```
   setenv PEX_FMT_HP_PORT_MAP_MODE TEMPLATE
   ```

2. Run all three stages of the Calibre xRC tool.

   The third stage, the formatter, searches first the SVDB directory and then the working directory for previously-defined template files. For any cell without a template file, the formatter generates a default template. The formatter derives the default template's contents from a cell's interface description contained in the Layout Netlist.

3. Make any necessary corrections for pin order in the template files.
4. Run the third stage again.

# Joining a Disjoint Parasitic Model

Use the PEX_FMT_CHECK_NET_FRAGMENTS environment variable to join a disjoint parasitic model. When you set this variable to ON, 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 environment variable in specific limited cases where you require a “clean” LVS when using Virtual Connect specification statements for locally disjoint power or signal nets. Ultimately, your design should be LVS clean without using either Virtual Connect statements or this environment variable.

When you use this environment variable, 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.

If you want a listing of the connections the Calibre xRC formatter makes, then use the -fmt_warnings switch on the command line during Calibre xRC formatter invocation.

# Using Port Names for Net Names

The standard behavior is for the Calibre xRC formatter to use 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.

# Verifying Timing with Probe Points

The PEX Probe File SVRF statement allows you to verify timing from specific points on each net by setting up probe points. Within the parasitic model, each probe point exists as an internal connection with a unique *probe_name*. You will be able to access each probe point by net and *probe_name*, depending on your simulator’s behavior. For more information on the PEX Probe File SVRF statement, refer to the Standard Verification Rule Format Manual.

# Chapter 11
# Integration and Troubleshooting Topics



## 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 Socket Port for a Layout Viewer" appendix of the *Calibre Interactive User's Manual*.

### 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. These files are the following:

- 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 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" 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 11-1. Calibre Interactive Settings Sequence**

| Stage | Action |
|---|---|
| 1 | Load the preferences file specified by $MGC_CALIBRE_PEX_RUNSET_FILE. By default, this points to $HOME/.cgipexdb.<br><br>The preferences file populates the runset list and may also supply default field settings. |
| 2 | Load a runset file and fill in the fields in the GUI.<br><br>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. |
| 3 | Load files as user specifies.<br><br>Typically, the user loads at least one rule file but may also load additional runsets. *Settings are not loaded into the GUI until the user clicks Load.*<br><br>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. |
| 4 | At execution, Calibre Interactive prepares a control file and runs any pre- and post-trigger run scripts. |
| 5 | 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. |

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*.

## Requirements

- 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 being 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 users 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. For more details, see Application Note 10236, "Using Interactively Generated Rules in a Calibre Batch Run."

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 11-1 on page 151.)

## Task Validation

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 11-2. 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 SupportNet 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. | 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.<br><br>Because some users may have a local version of rm, we recommend setting it explicitly also as<br>`/bin/rm -rf`<br><br>or as appropriate for your operating system. |
| Set environment variables explicitly in the script rather than the user environment. | Setting environment variables explicitly in the script serves several purposes:<br>• Easier to verify when debugging.<br>• Reduces problems due to variables remaining set in the shell; they are only active while the script is executing.<br>• Less typing than setting in the shell each time.<br>• 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 11-1. 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 ######################
#########################################################################
#            Mentor Graphics 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

There are several methods to make Calibre xRC runs shorter. Not all of the following may apply to your situation.

- Use the -64 invocation switch.

  Some Linux environments support both 32- and 64-bit executables and default to 32-bit. In these situations, you may see a performance improvement by adding -64 to invocations. (If your hardware only supports 32-bit, Calibre detects this and ignores the setting.)

- 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*.

- Turn on ASIC optimizations.

  The ASIC optimizations are enabled by default for LEF/DEF and Milkyway input. For GDS layouts, you need to supply the -asic switch at the PDB stage. The ASIC optimizations can be useful for very large, regular designs with Manhattan geometry such as digital designs completed with auto place and route. (The software may disable ASIC optimizations if the layout does not meet its criteria.)

- 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" on page 137.

## 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 Appendix D, "Reduction Techniques".
- 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 or environment variable:

PEX Netlist Global Nets
PEX Netlist … SPEF
PEX Netlist … DSPF
PEX Netlist … HSPICE SHORTPINNAMES
PEX Netlist Noxref Net Names
PEX Netlist Create Smashed Device Names
PEX_FMT_SPF_MODEL_NAME_MODE

## 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 Mentor Graphics support personnel for initially diagnosing problems.

## Setting Up For Troubleshooting

Follow these steps when you get a call from a user who is having trouble.

### Requirements

- 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 Appendix 13, "Calibre xRC Tool Invocation Reference".
- 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 SupportNet.

# Chapter 12
# Handling Parasitic On-Chip Variation



## What Is 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.

### What are the Sources of On-Chip Variation?

There are several sources:

- 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.

## How does Calibre Parasitic Extraction Model On-Chip Variation?

The Calibre parasitic extraction software provides three 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" on page 160 for more detail.
- Process corners. By providing the variations you need for your process corners, you can perform a single extraction run to create netlists for each of the corners. See "Process Corners" on page 161 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.

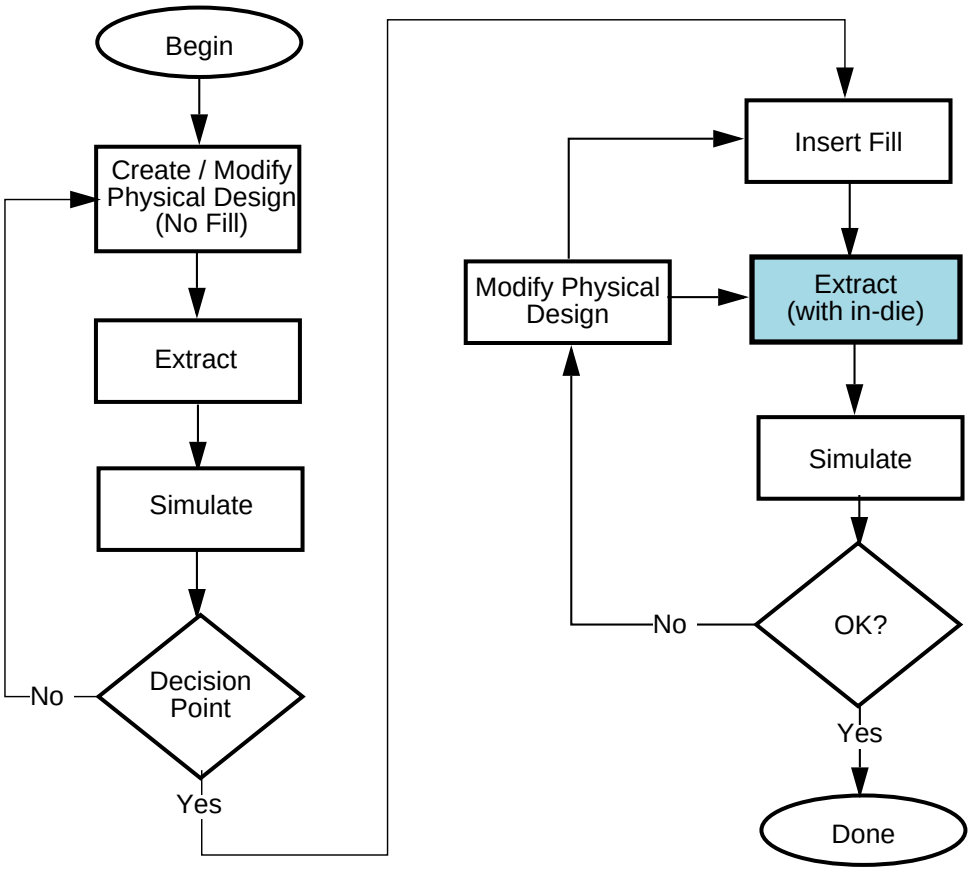
## When Should I Use On-Chip Variation During Parasitic Extraction?

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 12-1.

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 12-1. Design Flow Showing In-Die**




# Parasitic Extraction Techniques for On-Chip Variation

This section explains what the following techniques are doing and how to deploy them:



The techniques are independent of each other and may be singly or jointly used in extraction.

# In-Die Variation

## Essentials

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 documented in "In-Die Variation Files" in the *xCalibrate Interactive User's Manual*.
3. The foundry runs calibration, producing calibrated rules to provide as part of its design kit.

   The calibrated rules include the environment variables which control in-die effects. The default variable has been XCAL_EXCL_INDIE.
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**

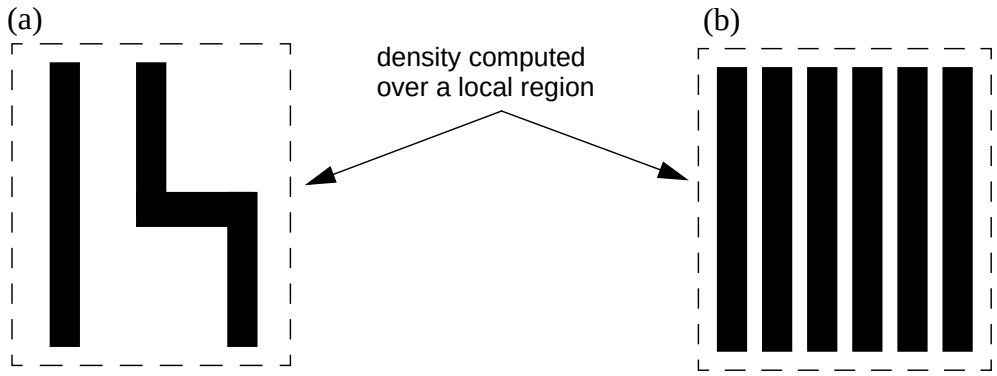
Older calibrated rule files may already include some Parasitic Variation statements.

### 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 12-2 shows how local density can vary from region to region.

**Figure 12-2. 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

### Essentials

Process corners can be created by both the foundry and the chip development team. When Calibre xRC extracts parasitics, all corners are extracted at once. You can choose to netlist all corners, or just a few.

Multiple corner extraction requires rules generated by xCalibrate v2008.2 or newer.

When using ASIC optimizations, only one corner can be extracted at a time.

### Method

1. The foundry sets up a calibration for multiple corners, as described in "Using the Multiple Corner Flow" in the *xCalibrate Batch User's Manual*. 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 Process Corners" on page 141 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.

When calibrating for multiple corners, the foundry uses data representing these best and worst variations. They run xCalibrate to produce rules which reflect this with variations in resistance, thickness, etc. Some of these variances are reflected in the PEX Table rules, which are used by the encrypted capacitance and resistance calculations.

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

### Essentials

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" on page 132 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 12-3. Typical Effects of CMP**

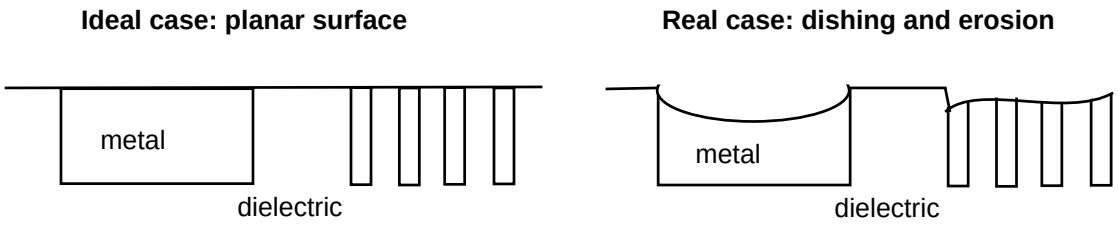
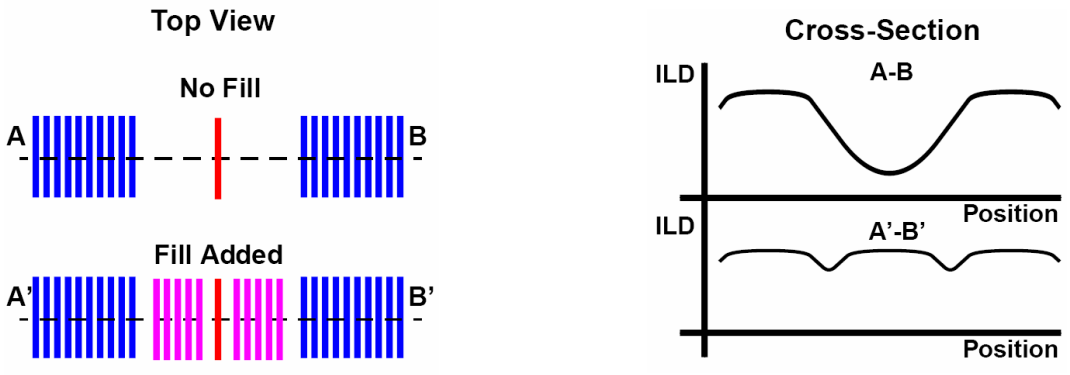



Figure 12-3 shows the potential effects of CMP. The metal is softer than the dielectric, and so is abraded more rapidly resulting in dishing. Where dense stretches of metal switch to dielectric, the polishing agent can "pile up" on one side, resulting in uneven erosion. Design geometry interacts with the CMP forces to dictate exactly how thickness will vary.

Figure 12-4 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 12-4. 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 13
# Calibre xRC Tool Invocation Reference

This chapter contains the following sections:



## Reference Syntax

The following notational elements are used in the syntax to indicate whether an argument is optional or required.

| | |
|---|---|
| **Bold** | A bold font indicates a required argument |
| [ ] | Square brackets enclose optional arguments. |
| *Italic* | An italic font indicates a user-supplied argument. |
| { } | Braces enclose arguments showing grouping. |
| \| | A vertical bar indicates an either/or choice between items |

**Note**

Omit braces, vertical bars, and notational fonts listed above when you actually enter a command.

# 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

This section provides reference pages for the Calibre xRC tool's command line invocation syntax. This section covers the following tool invocation commands:

- Source-Based connectivity and device recognition — "calibre -lvs" on page 167
- Layout-Based connectivity and device recognition — "calibre -xrc -phdb" on page 168
- Extraction and analysis of nets — "calibre -xrc -pdb" on page 170
- Format the netlist or report — "calibre -xrc -fmt" on page 173

# calibre -lvs

## Usage

**calibre -lvs -hier -spice *directory_path*/*layout_primary*.sp *rule_file_name***

## Description

Runs the Calibre nmLVS-H tool and creates the Persistent Hierarchical Database (PHDB) for use by the Calibre xRC tool.

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 SVRF statement in the rule file
- *layout_primary*.sp is the design's top-level cell you specified with the Layout Primary SVRF statement in the rule file

For complete Calibre nmLVS-H information, see the "Calibre nmLVS/nmLVS-H" 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.

## Example

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

## Usage

**calibre -xrc -phdb**
[-cb | {[-turbo [*nmbr_of_cpus*] [-turbo_all] }] [-hcell *hcell_list*] [-E *output*] [-64 | -32 ]
***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.

## Switches

- **-xrc**

  Performs parasitic extraction using the Calibre xRC tool. For the PHDB stage, the netlists will contain layout names. Use calibre -lvs for schematic names.

- **-phdb**

  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 command 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

  Runs connectivity extraction using xRC-CB licenses. Cannot be used with the -turbo switch. See “Extracting a Block Using CB” on page 106 for complete information.

- -turbo *nmbr_of_cpus*

  Specifies using multi-threaded parallel processing for PHDB creation. The *nmbr_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 switch, 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

  Optional argument used with the -turbo switch. This switch halts the invocation if it cannot secure the exact number of CPUs specified using -turbo.

- -hcell *hcell_list*

  Specifies the path to and name of the hcell file. It is used to create hierarchical PHDBs. See "Controlling Hierarchy with Xcells" on page 121 for an in-depth discussion of hcells and xcells.

- -E *output*

  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.

- -64 | -32

  Specifies using the 64-bit or 32-bit version. The default is to use the 64-bit version on Solaris operating systems that support it. For more information on 64-bit operations, refer to the *Calibre Administrator's Guide*.

- ***rule_file_name***

  Specifies the path to and name of the SVRF rule file.

## Example

The following command invokes PHDB creation using a Calibre xRC license:

```
calibre -xrc -phdb -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

## Usage

calibre **-xrc -pdb** [**-r** | **-c** | **-rc** | **-rcc** | **-adms**]
[-cb | {[-turbo [*nmbr_of_cpus*] [-turbo_all]] }]
[-xcell *xcell_list* [-incontext | -full]]
[-select | -cselect] [-asic | -noasic]
[-nocheck] [-pdb_info]
[-E *output*] [-64 | -32] ***rule_file_name***

## Description

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.

## Switches

- **-xrc**

  Performs parasitic extraction using the Calibre xRC tool.

- **-pdb**

  Performs parasitic extraction and analysis on data in the PHDB, and generate the PDB.

- [**-r** | **-c** | **-rc** | **-rcc** | **-adms**] (choose one)

  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” on page 70 for the preferred method. |

- -cb

  See “Extracting a Block Using CB” on page 106 for complete information.

- -turbo *nmbr_of_cpus*

  Specifies using multi-threaded parallel processing for PDB creation. The *nmbr_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 licences. If you do not apply the -turbo switch, 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

  Optional argument used with the -turbo switch. This switch halts the invocation if it cannot secure the exact number of CPUs specified using -turbo.

- -xcell *xcell_list*

  Optional switch to create the PDB hierarchically. 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 "Controlling Hierarchy with Xcells" on page 121.

  When -xcell is specified without -incontext or -full, the primary cell is extracted to the boundaries of the xcells.

- -incontext

  Optional switch used in conjunction with the -xcell switch. Extracts only those instances specified in the xcell list with a -C and a layout path. For more information, see "In-Context Extraction" on page 44.

- -full

  Optional switch used in conjunction with the -xcell switch. Performs hierarchical extraction, with fully netlisted xcells. For more information, see "Full Hierarchical Extraction" on page 40.

- -select | -cselect

  Specifies exclusively extracting nets using the net names you specify with PEX Extract Include SVRF statement. For more information, see "Extracting Particular Nets" on page 137.

  The -cselect switch can only be used for lumped capacitance (-c) 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.

- -asic | -noasic

  Specifies whether or not to use ASIC optimizations. By default, ASIC optimizations are on for LEF/DEF input, and off for other formats. ASIC optimizations should only be used with gate-level extraction. All capacitance is reported as intrinsic capacitance in -asic mode.

- -nocheck

  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

  Controls whether THRESHOLDING messages from the analyzer are printed to the transcript. If you add the switch to the command line, messages from the analyzer are printed to the transcript. If you do not use the switch, messages from the analyzer are not printed to the transcript.

- -E *output*

  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.

- -64 | -32

  Specifies using the 64-bit or 32-bit version. The default is to use the 64-bit version on Solaris operating systems that support it. For more information on 64-bit operations, see the *Calibre Administrator's Guide*.

- ***rule_file_name***

  Specifies the path to and name of the SVRF rule file.

## Example

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

## Usage

calibre **-xrc -fmt** [**-netmodel**] [**-c** | **-r** | **-rc** | **-rcc** | **-adms** | **-all** | **-simple**]
[-g] [-cb]
[-xcell *xcell_list* [-full | -incontext]]
[-corner {*corner*[,*corner*]… | all}]
[-fmt_warnings] [-fmt_info]
[-nocheck]
[-E *output*] [-64 | -32] ***rule_file_name***

## Description

Produces netlists and reports for the PDB's 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.

## Switches

- **-xrc**

  Specifies performing parasitic extraction using the Calibre xRC tool.

- **-fmt**

  Specifies producing netlists and reports from parasitic data stored in PDBs.

  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 switches 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**

  Specifies performing netlist formatting using a file used for selected nets with assigned net models. The SVRF rule file must contain a PEX Netlist Select 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**} (*choose one*)

  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” on page 70 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. The SVRF file must contain a PEX Netlist Simple statement. |

- -g

  Grounds any coupling capacitors.

- -cb

  For more information see “Extracting a Block Using CB” on page 106.

- -xcell *xcell_list*

  Formerly, required with all hierarchical PDBs to specify the path to and name of the xcell file. Retained for backwards compatibility in scripts.

- -full

  Specifies that the formatter should produce a fully hierarchical netlist.

- -incontext

  Sets the formatter to output only the contextualized cells, without the nets of the parent.

- -corner {*corner*[,*corner*]… | all}

  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 switch, 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

  Displays warning messages while the Calibre xRC formatter runs. Otherwise, messages are not displayed.

- -fmt_info

  Displays informational messages while the Calibre xRC formatter runs. Otherwise, messages are not displayed.

- -nocheck

  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*

  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.

- -64 | -32

  Specifies using the 64-bit or 32-bit version. The 64-bit version is the default on Solaris operating systems that support it. For more information on 64-bit operations, see the *Calibre Administrator's Guide*.

- ***rule_file_name***

  Specifies the path to and name of the SVRF rule file.

## Example

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 my_selected_nets
```

For more information on PEX Netlist Select see the *Standard Verification Rule Format (SVRF) Manual*.

# Chapter 14
# Environment Variables

You control specific tool-produced netlist formatting operations with the Calibre xRC tool using environment variables. The Calibre xRC formatter uses the majority of these environment variables for formatting the output netlist.

## Using Environment Variables

In general, you specify any environment variable you use in the UNIX or Linux shell from which you will invoke the Calibre xRC tool before running the tool.

For example, you would specify the PEX_DEF_OUTPUT_SOURCE_NAME environment variable in a C shell using the following syntax:

```
setenv PEX_DEF_OUTPUT_SOURCE_NAME ON
```

and subsequently invoke the Calibre xRC tool. In the Bourne family of shells (for example, bash and ksh) you would use the following syntax:

```
PEX_DEF_OUTPUT_SOURCE_NAME=ON; export PEX_DEF_OUTPUT_SOURCE_NAME
```

## Variables By Functionality

The environment variables are grouped in the tables below according to functionality. The tables represent only major functional groupings; not all environment variables are listed in them. To look up a variable by name, see the variable dictionary starting at page 184.

### Analysis and Extraction Related



### Netlisting and Output Related

**Table 14-1. Variables for LEF/DEF Input**

| Environment Variable | Default |
|---|---|
| PEX_DEF_EXTRACT_MACRO_OBS | unset |
| PEX_DEF_EXTRACT_METAL_FILLS | on |
| PEX_DEF_MAP | unset |
| PEX_DEF_OUTPUT_SOURCE_NAME | unset |

**Table 14-2. Variables for Net Content**

| Environment Variable | Default |
|---|---|
| PEX_EXTRACT_FLOATING_NETS | off |
| PEX_FMT_GLOBAL | unset |
| PEX_FMT_MIN_CAP_VALUE | unset |
| PEX_FMT_NOXREF_MODEL_MODE | KEEP |
| PEX_FMT_RC_NAMED_PARAMETER | OFF |
| PEX_FMT_UNSHORT_DEVICE_PIN_MODE_R | 0.01 ohm |
| PEX_NASSDA | off |

**Table 14-3. Variables for Tailoring HSPICE Netlists**

| Environment Variable | Default |
|---|---|
| PEX_FMT_ESCAPE_CHARS | “!@#$%^&*()+{}\|\:;’‘<>?/.,-=” |
| PEX_FMT_GLOBAL | unset |
| PEX_FMT_HSPICE_NAME_FILTER_MODE | no filtering |
| PEX_FMT_R_MODEL | ideal resistance |
| PEX_FMT_SPICE_KEYWORD_UPCASE | off |
| PEX_FMT_SPICE_PIN_SEPARATOR | “_” |
| PEX_FMT_SPICE_USE_SHORT_NAMES | off |

**Table 14-4. Variables for Tailoring SPEF Netlists**

| Environment Variable | Default |
|---|---|
| PEX_FMT_ESCAPE_CHARS | “!@#$%^&*()+{}\|\:;’‘<>?/.,-=” |
| PEX_FMT_SPEF_BUS_DELIMITER | "" |
| PEX_FMT_SPEF_LAYER_MAP | off |
| PEX_FMT_SPEF_NAME_FILTER_MODE | no filtering |
| PEX_FMT_SPEF_NAME_MAP | off |
| PEX_FMT_SPF_IGN_SMASHED_NAME | off |
| PEX_FMT_SPF_LUMPED_MODEL_MODE | instance |

**Table 14-5. Variables for Tailoring DSPF Netlists**

| Environment Variable | Default |
|---|---|
| PEX_FMT_ESCAPE_CHARS | “!@#$%^&*()+{}\|\:;’‘<>?/.,-=” |
| PEX_FMT_SPF_NAME_FILTER_MODE | no filtering |
| PEX_FMT_SPF_IGN_SMASHED_NAME | off |
| PEX_FMT_SPF_INSTANCE_SECTION | off |
| PEX_FMT_SPF_LUMPED_MODEL_MODE | instance |
| PEX_FMT_SPF_MODEL_NAME_MODE | by net name |
| PEX_FMT_SUPPRESS_DSPF_SUBCKT | unset |
| PEX_FMT_UNSHORT_DEVICE_PIN_MODE_R | off<br>0.01 resistor between shorted pins |

**Table 14-6. Variables for Controlling Names**

| Environment Variable | Default |
|---|---|
| PEX_FMT_SOURCE_BASED_FLOW | OFF |
| PEX_PORTS_MARK_SIGNALS | ON |

# CALIBRE_ECHO_RULE_FILE

## Usage

CALIBRE_ECHO_RULE_FILE ON

## Description

Sets Calibre to echo all processed statements to standard out instead of only statements in the top SVRF file.

Use this variable when debugging SVRF files to see the result of conditionals and included files. When CALIBRE_ECHO_RULE_FILE is set, all statements used are echoed.

**Note**

By default, the variable is not set. When it is set, opening Calibre® RVE™ from Calibre® DESIGNrev™ is significantly slower because all rules are echoed to standard out. To prevent this, unset the variable before invoking Calibre DESIGNrev.

## Examples

To echo all rules processed for a run:

```
setenv CALIBRE_ECHO_RULE_FILE ON
```

To restore default behavior:

```
unsetenv CALIBRE_ECHO_RULE_FILE
```

# PEX_CMP_FILE

**Note**

This environment variable has been deprecated as of the 2009.1 release. Use the PEX CMP Mode SVRF statement instead.

## Usage

PEX_CMP_FILE ***data_location***

## Description

Points to the data for chemical mechanical polishing (CMP) models used for thickness modeling.

**Note**

If this variable has been set and you want to use in-die variation or nominal thickness instead, the variable must be unset.

For the default Praesagus CMP format, ***data_location*** must be a file.

For VCMP format, ***data_location*** must be a directory.

## Examples

To set the directory *cmp_out* as the file location:

```
setenv PEX_CMP_FILE $HOME/model1/cmp_out
```

To use other thickness modeling methods, you must unset this variable, for example:

```
unsetenv PEX_CMP_FILE
```

## Related Variables and SVRF Statements

PEX_CMP_MODE

PEX_CMP_SUFFIX

# PEX_CMP_MODE

**Note**

This environment variable has been deprecated as of the 2009.1 release. Use the PEX CMP Mode SVRF statement instead.

## Usage

PEX_CMP_MODE **[praesagus | VCMP]**

## Description

Specifies which CMP format the data is in. The default is Praesagus' Copper Prediction and Verification database. The other supported format is TSMC's Virtual Chemical Mechanical Polishing (VCMP) format.

## Examples

To specify a Praesagus database:

```
setenv PEX_CMP_MODE praesagus
```

To specify TSMC's VCMP format:

```
setenv PEX_CMP_MODE VCMP
```

## Related Variables and SVRF Statements

PEX_CMP_FILE

PEX_CMP_SUFFIX

# PEX_CMP_SUFFIX

**Note**

This environment variable has been deprecated as of the 2009.1 release. Use the PEX CMP Mode SVRF statement instead

## Usage

PEX_CMP_SUFFIX ***string***

## Description

Specifies the suffix used by the VCMP files. The VCMP format defaults to “z2o.txt”. Ignored for Praesagus CMP.

The Calibre xRC tool reads VCMP files named ***layer****z2o.txt*, or more generally ***layerstring***, for each layer. The layer name is forced into all lowercase for the file name.

## Examples

For a design containing layers Poly, M1, and POD, the corresponding VCMP files are *poly.cmp.txt*, *m1.cmp.txt*, and *pod.cmp.txt*. The following example sets the suffix so that the Calibre xRC tool correctly identifies the files:

```
setenv PEX_CMP_MODE VCMP
setenv PEX_CMP_FILE $HOME/65nm/cmp_data
setenv PEX_CMP_SUFFIX .cmp.txt
```

## Related Variables and SVRF Statements

PEX_CMP_FILE

PEX_CMP_MODE

# PEX_DEF_EXTRACT_MACRO_OBS

**Note**

PEX_DEF_EXTRACT_MACRO_OBS environment variable has been deprecated as of the 2008.3 release. It has been replaced by the PEX DEF Extract Cell Obstructions SVRF statement.

## Usage

PEX_DEF_EXTRACT_MACRO_OBS {OFF | ON}

## Description

Includes obstruction geometries when extracting parasitics. By default, obstructions are ignored. Setting this variable will change the extracted parasitics, particularly for parasitic capacitance. The geometries are not included in the netlist written out by the formatter.

## Examples

To include obstruction geometries, enter the following before running Calibre xRC:

```
setenv PEX_DEF_EXTRACT_MACRO_OBS ON
```

To return to the default of ignoring obstruction geometries, enter the following before running Calibre xRC:

```
setenv PEX_DEF_EXTRACT_MACRO_OBS OFF
```

# PEX_DEF_EXTRACT_METAL_FILLS

## Usage

PEX_DEF_EXTRACT_METAL_FILLS {OFF | ON}

## Description

Includes metal fills specified with LEF 5.6 format when extracting parasitics. By default, metal fill is included.

## Examples

To include metal fill geometries, enter the following before running Calibre xRC:

```
setenv PEX_DEF_EXTRACT_METAL_FILLS ON
```

## Related Variables and SVRF Statements

Table 14-1, “Variables for LEF/DEF Input” on page 178

PEX Extract Floating Nets in the *Standard Verification Rule Format (SVRF) Manual.*

# PEX_DEF_MAP

**Note**

The PEX_DEF_MAP environment variable has been deprecated as of the 2008.3 release. The functionality has been replaced with the Layer SVRF statement.

## Usage

PEX_DEF_MAP ***mapfile***

## Description

Specifies a file which maps LEF/DEF layer names to layer names used in the SVRF rule file. The default is no mapping file.

The mapping file's entries must be in the form

```
DEF_LAYER_MAP def_layer_name rule_layer_name
```

The parameter ***mapfile*** can be either a relative path such as *../../def_map_file.txt* or an absolute path. Layer names are case sensitive.

## Examples

At the command line or the Calibre Interactive environment window, type the following:

```
setenv PEX_DEF_MAP ./def_mapping_file.txt
```

The mapping file, def_mapping_file.txt, would contain entries such as the following:

```
DEF_LAYER_MAP m1 metal1
DEF_LAYER_MAP m2 metal2
```

# PEX_DEF_OUTPUT_SOURCE_NAME

## Usage

PEX_DEF_OUTPUT_SOURCE_NAME {OFF | ON}

## Description

For LEF/DEF designs when diagnosing short circuits, use the source name instead of a number in the layout net. Underscores (_) replace slashes (/) to produce valid SPICE names. The default is to use the net number.

Use this variable to produce more readable net names when tracking down short circuits in a design.

The netlist produced by the formatter for LEF/DEF input always contains source names. This variable does not affect those netlists.

## Examples

For a DEF file that defines net 5 as “example/netA”, the following net names would be shown when tracking down a short:

| | |
|---|---|
| `setenv PEX_DEF_OUTPUT_SOURCE_NAME ON` | produces `example_netA` |
| `setenv PEX_DEF_OUTPUT_SOURCE_NAME OFF` | produces `5` |

## Related Variables and SVRF Statements

Table 14-1, “Variables for LEF/DEF Input” on page 178

# PEX_EDGE_BASED_SPACING

**Note**

This environment variable is deprecated as of the 2009.1 release. Use the PEX Indie Spacing SVRF statement instead.

## Usage

PEX_EDGE_BASED_SPACING {**ON** | **OFF**}

## Description

For in-die variation, controls whether spacing is based on edges or overall density. The default is edge-based (variable is on).

Edge-based spacing calculates the in-die variation in width of a wire based upon the spacing from the wire's drawn edge to the edges of its nearest neighbors on the same layer.

The actual thickness of a line is macroscopic in nature and is a function of the local density of other shapes in its immediate vicinity. SVRF statements such as PEX Table and Parasitic Variation compensate for in-die variations in width and thickness when calculating parasitic resistance and capacitance values.

**Note**

In version 2007.2, edge-based spacing became the default basis for in-die calculations. Prior to version 2007.2, the default setting averaged the distance to the neighbors on both sides.

## Examples

To restore pre-2007.2 calculations:

```
setenv PEX_EDGE_BASED_SPACING OFF
```

# PEX_EXTRACT_FLOATING_NETS

**Note**

This environment variable is deprecated as of the 2007.4 release. It has been replaced with the PEX Extract Floating Nets SVRF statement.

## Usage

PEX_EXTRACT_FLOATING_NETS {OFF | ON}

## Description

Set to extract floating nets from the layout. The default is off, or ignore floating nets.

See "Ignoring or Extracting Floating Nets" on page 139 for a discussion of this variable's use.

## Examples

```
setenv PEX_EXTRACT_FLOATING_NETS OFF
```

## Related Variables and SVRF Statements

PEX_FLOATING_NET_COUPLING

PEX Extract Floating Nets

# PEX_FDI_MAP

## Usage

PEX_FDI_MAP ***mapfile***

## Description

Specifies a file which maps FDI layer names to layer names used in the SVRF rule file. The default is no mapping file.

The mapping file's entries must be in the form

```
FDI_LAYER_MAP fdi_layer_name rule_layer_name
```

The parameter ***mapfile*** can be either a relative path such as *../../def_map_file.txt* or an absolute path. Layer names are case sensitive.

## Examples

At the command line type the following:

```
setenv PEX_FDI_MAP ./fdi_mapping_file.txt
```

The mapping file, fdi_mapping_file.txt, would contain entries such as the following:

```
FDI_LAYER_MAP met1 metal1
FDI_LAYER_MAP met2 metal2
```

# PEX_FLOATING_NET_COUPLING

**Note**

The environment variable is deprecated as of the 2007.4 release. Use the PEX Extract Floating Nets REDUCE SVRF statement instead.

## Usage

PEX_FLOATING_NET_COUPLING {OFF | ON}

## Description

Enables floating net coupling algorithm to more accurately extract coupled capacitance between signal nets and non-extracted floating nets, such as metal fill. Default is off.

The algorithm requires that there are signal nets and floating nets in the design.

## Examples

```
setenv PEX_FLOATING_NET_COUPLING ON
```

## Related Variables and SVRF Statements

PEX_PORTS_MARK_SIGNALS

PEX Extract Floating Nets

# PEX_FMT_CHECK_NET_FRAGMENTS

**Note**

This variable is deprecated as of the 2009.1 release. Use the PEX Netlist Virtual Connects SVRF statement instead.

## Usage

PEX_FMT_CHECK_NET_FRAGMENTS {**OFF** | **ON**}

## Description

Connects disjoint net model fragments created by Virtual Connect statements to the main net model using a resistor. The default is to not connect fragments.

The resistor value defaults to zero. You can set a different value using the environment variable PEX_FMT_UNSHORT_DEVICE_PIN_MODE_R.

You should only use this environment variable in specific, limited cases, usually when testing subsets of the design. Ultimately, your design should be LVS clean without using this environment variable.

**Caution**

The nodes used to connect the fragment to the net model are essentially arbitrary. Do not use this variable during final sign-off tests.

To see a list of the connections made by the formatter, use the -fmt_warnings option.

## Examples

To attach floating pins while the root cause is being identified:

```
setenv PEX_FMT_CHECK_NET_FRAGMENTS_ON
setenv PEX_FMT_UNSHORT_DEVICE_PIN_MODE_R 0.01
```

## Related Variables and SVRF Statements

"PEX_FMT_UNSHORT_DEVICE_PIN_MODE_R" on page 221

Virtual Connect in the *Standard Verification Rule Format (SVRF) Manual*.

"calibre -xrc -fmt" on page 173

# PEX_FMT_ESCAPE_CHARS

**Note**

This variable is deprecates as of the 2009.1 release. Use the PEX Netlist Escape Characters SVRF statement instead.

## Usage

PEX_FMT_ESCAPE_CHARS "***string***"

## Description

This variable specifies escape characters for use in the output file. You must enclose the string within quotation marks.

The default value is !@#$%^&*()+{}|\:;’‘<>?/.,-=

## Examples

To restore the variable’s default value:

```
setenv PEX_FMT_ESCAPE_CHARS "!@#$%^&*()+{}|\:;’‘<>?/.,-="
```

## Related Variables and SVRF Statements

Table 14-3, “Variables for Tailoring HSPICE Netlists” on page 178

Table 14-4, “Variables for Tailoring SPEF Netlists” on page 179

# PEX_FMT_GLOBAL

**Note**

This variable is deprecated as of the 2009.1 release. Use the PEX Netlist Global Nets SVRF statement instead.

## Usage

PEX_FMT_GLOBAL "***name1; name2; namen”***

## Description

PEX_FMT_GLOBAL removes the named nets from the subcircuit and places them in a global name definition. Default is unset.

When you set this variable with global power or ground net names, the Calibre xRC formatter removes the nets from the pinlist of the instantiated subcircuit and places them in a global name definition. This makes HSPICE netlists more compact.

**Note**

Global nets must also be listed in the PEX Extract Exclude statement. You cannot netlist parasitics on global nets in hierarchical extraction.

When setting the variable for multiple net names, separate the names with a semicolon (;) and no intervening spaces.

## Examples

To place VSS and VDD in the global name definition:

```
setenv PEX_FMT_GLOBAL "VSS;VDD"
```

To restore the default value:

```
unsetenv PEX_FMT_GLOBAL
```

# PEX_FMT_HP_PORT_MAP_MODE

## Usage

PEX_FMT_HP_PORT_MAP_MODE “{TEMPLATE | TEMPLATE CELLNAME SOURCE}“

## Description

Enables templates. Templates can be used for re-ordering pins and specifying pin direction in netlists. The default is for the variable to not be set (no templates).

**Note**

For re-ordering pins to match schematic-based test benches, the PEX Pin Order command is preferable. It does not handle intermediate circuits or pin direction, however.

Setting the variable to TEMPLATE maps ports using the user-defined template for each cell. TEMPLATE CELLNAME SOURCE defines an alternate name for the cell in the netlist. Either of these settings cause the PEX Pin Order statement to have no effect.

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*). The formatter derives the default template’s contents from a cell's interface description contained in the layout netlist.

For details on the template syntax, see “Templates” on page 264.

## Related Variables and SVRF Statements

PEX Pin Order in the *Standard Verification Rule Format (SVRF) Manual.*

# PEX_FMT_HSPICE_NAME_FILTER_MODE

**Note**

As of version 2008.4, this variable is deprecated and replaced by the NOINSTANCEX keyword in the PEX Netlist… HSPICE statement.

## Usage

PEX_FMT_HSPICE_NAME_FILTER_MODE [X]

## Description

Causes formatter to filter Xs in HSPICE output. Set this variable by specifying X.

## Related Variables and SVRF Statements

Table 14-3, “Variables for Tailoring HSPICE Netlists” on page 178

# PEX_FMT_MIN_CAP_VALUE

## Usage

PEX_FMT_MIN_CAP_VALUE ***pFs***

## Description

**Note**

The PEX_FMT_MIN_CAP_VALUE environment variable is deprecated as of the 2007.4 release. It has been replaced by the PEX Reduce Mincap SVRF statement. When PEX Reduce Mincap REMOVE is used in a rule file while PEX_FMT_MIN_CAP_VALUE is set, the Remove value set by the SVRF statement overrides the environment variable.

Prevents capacitors below the specified threshold from being written to the extraction netlist. It is applied after other forms of reduction.

**Note**

This option *removes* the capacitors; they are not added to other capacitance values, as is done in other forms of reduction. Timing is not preserved, and you will need to compensate for the removed capacitors.

## Examples

The following setting removes parasitic capacitors that are less than 0.001 picofarad (that is, less than 1 femtofarad).

```
setenv PEX_FMT_MIN_CAP_VALUE 0.001
```

## Related Variables and SVRF Statements

PEX Reduce Mincap in the *Standard Verification Rule Format (SVRF) Manual.*

# PEX_FMT_NOXREF_MODEL_MODE

**Note**

This variable has been deprecated as of the 2009.1 release. Use the PEX Netlist Noxref Net Names SVRF statement instead.

## Usage

PEX_FMT_NOXREF_MODEL_MODE {**KEEP** | **NONE**}

## Description

Keeps or eliminates parasitic net models marked with _noxref from netlists. Default is keep. Nets with “noxref” in their name still appear in the intentional circuitry and are treated as ideal nets.

When you set this variable to NONE, the formatter suppresses parasitic net models for the _noxref nets from the netlist. These nets are often ones inside a cell recognized as a gate by LVS, but extracted by xRC.

For more information on _noxref nets, see “Interpreting _noxref Entries” on page 261.

## Examples

For a design with one intentional resistor in the schematic, but two in series in the layout, setting the PEX_FMT_NOXREF_MODEL_MODE variable affects the output of the simple HSPICE netlist as shown below:

**KEEP**

```
% grep -n -i subckt ./simple.rcc.pex
8:.subckt PM_SIMPLE%X1/4 1 6 8 9 12
14 15 17
26:.subckt PM_SIMPLE%X2/4 1 6 8 9 12
14 15 17
44:.subckt PM_SIMPLE%A 3 5
50:.subckt PM_SIMPLE%noxref_2 2 3
56:.subckt PM_SIMPLE%B 2 5
61:.subckt PM_SIMPLE%VDD 1 4 9 10 12
    14 16
75:.subckt PM_SIMPLE%4 1 2 4 11
83:.subckt PM_SIMPLE%GND 1 4 12 13
93:.subckt PM_SIMPLE%6 1 2 4 11
```

**NONE**

```
% grep -n -i subckt ./simple.rcc.pex
8:.subckt PM_SIMPLE%X1/4 1 6 8 9 12
14 15 17
26:.subckt PM_SIMPLE%X2/4 1 6 8 9 12
14 15 17
44:.subckt PM_SIMPLE%A 3 5
50:.subckt PM_SIMPLE%B 2 5
55:.subckt PM_SIMPLE%VDD 1 4 9 10 12
    14 16
69:.subckt PM_SIMPLE%4 1 2 4 11
77:.subckt PM_SIMPLE%GND 1 4 12 13
87:.subckt PM_SIMPLE%6 1 2 4 11
```

The subcircuit in line 50 of the KEEP column does not appear in the NONE column because it is suppressed. This subcircuit is a parasitic net model for the noxref_2 net as shown by the name “PM_SIMPLE%noxref_2”.

# PEX_FMT_R_MODEL

**Note**

This variable is deprecated as of the 2009.1 release. Use the PEX Netlist Device Resistance Model SVRF statement instead.

## Usage

PEX_FMT_R_MODEL [WIRE]

## Description

Specify wire model for use in HSPICE instead of ideal resistance.

When you specify this variable with WIRE, HSPICE uses a wire model for modeling parasitic resistance. Specifically, it uses the W and L parameters calculated within Device specification statements in the rule file. By default, HSPICE simulators model resistors as ideal resistances using HSPICE built-in models.

## Related Variables and SVRF Statements

Table 14-3, “Variables for Tailoring HSPICE Netlists” on page 178

# PEX_FMT_RC_NAMED_PARAMETER

## Usage

PEX_FMT_RC_NAMED_PARAMETER {**OFF** | **ON**}

## Description

Set to ON to replace R or C values with parameters for intentional devices values for ELDO, HSPICE, or DSPF netlists. For example, instead of “0.24” an entry will show “R=0.24”.

**Note**

The PEX_FMT_RC_NAMED_PARAMETER environment variable is deprecated as of the 2006.4 release. It has been replaced by the RCNAMED keyword in the PEX Netlist SVRF statement. The RCNAMED keyword overrides the variable’s setting.

## Related Variables and SVRF Statements

PEX Netlist in the *Standard Verification Rule Format (SVRF) Manual.*

PEX Netlist ADMS in the *Standard Verification Rule Format (SVRF) Manual.*

PEX Netlist Simple in the *Standard Verification Rule Format (SVRF) Manual.*

# PEX_FMT_SOURCE_BASED_FLOW

## Usage

PEX_FMT_SOURCE_BASED_FLOW {**OFF** | **ON**}

## Description

**Note**

As of version 2008.4, this variable is deprecated and replaced by the SOURCEBASED keyword in the PEX Netlist statement.

Activates source-based flow. *Supported with flat and gate-level extraction only.*

When this variable is set to ON, the Calibre xRC formatter uses the sub-circuit pin order and names in the source netlist instead of the layout netlist when it outputs the extracted netlist. If you are creating the source netlist with the Calibre nmLVS software, you must use nmLVS-H.

In all cases, the design must pass LVS. If the design still has LVS errors then when you run the formatter it will exit with the following error message:

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

Besides preserving the schematic pin order, source-based flow also uses the exact device names of the schematic rather than the form stored in the PHDB. (The schematic names and names stored in the PHDB may differ if the PHDB was created with flat LVS.) NETLIST MODEL and NETLIST ELEMENT settings in the Device SVRF statement are ignored.

# PEX_FMT_SPECTRE_NAME_FILTER

## Usage

PEX_FMT_SPECTRE_NAME_FILTER [X]

## Description

When you set this variable to ON by specifying X, the formatter filters instance names from SPECTRE output. It removes leading Xs from instance names.

This variable replaces PEX_FMT_SPECTRE_NAME_FILTER_MODE which was too long for some versions of Linux.

**Note**

As of version 2008.4, this variable is deprecated and replaced by the NOINSTANCEX keyword for the PEX Netlist… SPECTRE statement.

# PEX_FMT_SPECTRE_NAME_FILTER_MODE

## Usage

PEX_FMT_SPECTRE_NAME_FILTER_MODE [X]

## Description

When you set this variable to ON by specifying X, the formatter filters instance names from SPECTRE output. It removes leading Xs from instance names.

**Note**

As of version 2008.4, this variable is deprecated and replaced by the NOINSTANCEX keyword for the PEX Netlist… SPECTRE statement.

# PEX_FMT_SPEF_BUS_DELIMITER

## Usage

PEX_FMT_SPEF_BUS_DELIMITER *string*

## Description

Specifies an optional string to set the BUS_DELIMITER variable to in the SPEF header. The default is a pair of enclosing quotes (“”).

**Note**

As of version 2008.4, this variable is deprecated and replaced by the BUSDELIM keyword in the PEX Netlist… SPEF statement.

## Related Variables and SVRF Statements

Table 14-4, “Variables for Tailoring SPEF Netlists” on page 179

# PEX_FMT_SPEF_LAYER_MAP

## Usage

**PEX_FMT_SPEF_LAYER_MAP** {**OFF** | **ON**}

## Description

Specifies to output the layer information for each resistor as an integer and a commented layer map. Only affects SPEF-format netlists.

## Examples

When the variable is not set, the SPEF netlist looks as follows:

```
*RES
8 ENL:30 ENL:34 0.525075 // $lvl=M1
9 ENL ENL:30 0.07 // $lvl=M1
```

If the variable is set, for example using

```
setenv PEX_FMT_SPEF_LAYER_MAP ON
```

the SPEF netlist looks as follows:

```
// *LAYER_MAP

// *1 POLY
// *2 CONT
// *3 M1

...

*RES
8 ENL:30 ENL:34 0.525075 // $lvl=3
9 ENL ENL:30 0.07 // $lvl=3
.
```

## Related Variables and SVRF Statements

Table 14-4, “Variables for Tailoring SPEF Netlists” on page 179

# PEX_FMT_SPEF_NAME_FILTER_MODE

## Usage

PEX_FMT_SPEF_NAME_FILTER_MODE [X]

## Description

When you set this variable to ON by specifying X, the formatter filters instance names from SPEF output. It removes leading Xs from instance names. This is useful for non-SPICE analysis tools and for backannotation to Verilog and VHDL.

If you specify SPEF PRIMETIME in the PEX Netlist statement, this variable is automatically set.

**Note**

As of version 2008.4, this variable is deprecated and replaced by the NOINSTANCEX keyword in the PEX Netlist…SPEF statement.

## Related Variables and SVRF Statements

Table 14-4, “Variables for Tailoring SPEF Netlists” on page 179

# PEX_FMT_SPEF_NAME_MAP

## Usage

PEX_FMT_SPEF_NAME_MAP

## Description

When you set this variable to ON, the formatter generates a name map section in the SPEF netlist. The name map is a list that maps instance names to numerical identifiers. The identifiers are printed in place of the instance names in the netlist's model sections; this produces more compact SPEF netlists.

**Note**

As of version 2008.4, this variable is deprecated and replaced by the MAPNAMES keyword in the PEX Netlist… SPEF statement.

## Related Variables and SVRF Statements

Table 14-4, "Variables for Tailoring SPEF Netlists" on page 179

# PEX_FMT_SPF_IGN_SMASHED_NAME

**Note**

This variable is deprecated as of the 2009.1 release. Use the PEX Netlist Create Smashed Device Names SVRF statement instead.

## Usage

PEX_FMT_SPF_IGN_SMASHED_NAME [ON]

## Description

For both DSPF and SPEF output. When you set this variable to ON, the formatter does not create new instance names for smashed devices. The default is to create the new instance names.

This variable replaces PEX_FMT_SPF_IGNORE_INSTANCE_NAME_SMASHING which was too long for some versions of Linux.

## Related Variables and SVRF Statements

Table 14-4, “Variables for Tailoring SPEF Netlists” on page 179

Table 14-5, “Variables for Tailoring DSPF Netlists” on page 179

# PEX_FMT_SPF_IGNORE_INSTANCE_NAME_SMASHING

## Usage

PEX_FMT_SPF_IGNORE_INSTANCE_NAME_SMASHING

## Description

**Note**

As of version 2007.2, this variable is deprecated in favor of PEX_FMT_SPF_IGN_SMASHED_NAME.

For both DSPF and SPEF output. When you set this variable to ON, the formatter does not create new instance names for smashed devices. The default is to create the new instance names.

## Related Variables and SVRF Statements

Table 14-4, “Variables for Tailoring SPEF Netlists” on page 179

Table 14-5, “Variables for Tailoring DSPF Netlists” on page 179

# PEX_FMT_SPF_INSTANCE_SECTION

## Usage

PEX_FMT_SPF_INSTANCE_SECTION {**ON** | OFF}

## Description

When you set this variable to OFF, the formatter eliminates the instance section from DSPF output. This saves execution time in cases where the instance section is not needed, such as backannotation to Verilog.

The default behavior is to include the instance section in DSPF netlists.

**Note**

As of version 2008.4, this variable is deprecated and replaced by the NOINSTANCESECTION keyword in the PEX Netlist… DSPF statement.

## Related Variables and SVRF Statements

Table 14-5, “Variables for Tailoring DSPF Netlists” on page 179

# PEX_FMT_SPF_LUMPED_MODEL_MODE

## Usage

PEX_FMT_SPF_LUMPED_MODEL_MODE [NET | NONE]

## Description

Controls where the formatter writes the parasitics for lumped models. The default is to write a lumped capacitor in the Instance section, with an empty Net section.

When you set this variable to NET, the Calibre xRC formatter produces a lumped C netlist model compatible with TimeMill, PathMill, and PrimeTime. The netlist model consists of an entry in the Net section with a total C value and with no parasitic components, ports, or pins.

When you set this variable to NONE, the Calibre xRC formatter does not output lumped models. The netlist is compatible with PrimeTime.

When the variable is not set, the formatter produces a lumped capacitor in the Instance section with an empty Net section.

## Examples

To restore the default behavior:

```
unsetenv PEX_FMT_SPF_LUMPED_MODEL_MODE
```

To write the model in the Net section instead of the Instance section:

```
setenv PEX_FMT_SPF_LUMPED_MODEL_MODE NET
```

## Related Variables and SVRF Statements

Table 14-5, “Variables for Tailoring DSPF Netlists” on page 179

# PEX_FMT_SPF_MODEL_NAME_MODE

## Usage

PEX_FMT_SPF_MODEL_NAME_MODE [ID]

## Description

When you specify this statement with “ID”, the formatter produces a smaller DSPF netlist by not qualifying the names of parasitic resistors and capacitors using the net name as a prefix. Instead, it uses the net's ID number.

You can restore the default behavior of using the net name by either unsetting the variable, or setting it without ID, for example:

```
setenv PEX_FMT_SPF_MODEL_NAME_MODE
```

## Related Variables and SVRF Statements

Table 14-5, “Variables for Tailoring DSPF Netlists” on page 179

# PEX_FMT_SPF_NAME_FILTER_MODE

## Usage

PEX_FMT_SPF_NAME_FILTER_MODE [X]

## Description

When you set this variable to ON by specifying X, the formatter filters instance names from DSPF output. It removes leading Xs from instance names. This is useful for non-SPICE analysis tools and for backannotation to Verilog and VHDL.

**Note**

As of version 2008.4, this variable is deprecated and replaced by the NOINSTANCEX keyword in the PEX Netlist…DSPF statement.

## Related Variables and SVRF Statements

Table 14-5, “Variables for Tailoring DSPF Netlists” on page 179

# PEX_FMT_SPICE_KEYWORD_UPCASE

**Note**

This variable is deprecated as of the 2009.1 release. Use the PEX Netlist Uppercase Keywords SVRF statement instead.

## Usage

PEX_FMT_SPICE_KEYWORD_UPCASE ON

## Description

Output element names and HSPICE and DSPF keywords in upper case. Default behavior is to produce lower case text.

## Examples

To produce elements and keywords in capital letters:

```
setenv PEX_FMT_SPICE_KEYWORD_UPCASE ON
```

To return to the default setting:

```
unsetenv PEX_FMT_SPICE_KEYWORD_UPCASE
```

## Related Variables and SVRF Statements

Table 14-3, “Variables for Tailoring HSPICE Netlists” on page 178

# PEX_FMT_SPICE_NET_NAME_SEPARATOR

**Note**

As of version 2008.4, this variable is deprecated and replaced by the PINNAMESEP keyword in the PEX Netlist…HSPICE statement.

## Usage

PEX_FMT_SPICE_NET_NAME_SEPARATOR "***character***"

## Description

Defines the character which the formatter places between device pins and parasitic model subcircuits when creating net names. The character must occur between quotation marks (“ ”). The default is “_”.

## Examples

To use an underbar character as the net separator:

```
setenv PEX_FMT_SPICE_NET_NAME_SEPARATOR "_"
```

## Related Variables and SVRF Statements

Table 14-3, “Variables for Tailoring HSPICE Netlists” on page 178

# PEX_FMT_SPICE_PIN_SEPARATOR

**Note**

As of version 2008.4, this variable is deprecated and replaced by the PINNAMESEP keyword in the PEX Netlist…HSPICE statement.

## Usage

PEX_FMT_SPICE_PIN_SEPARATOR "***character***"

## Description

Defines the character which the formatter places between device pins and parasitic model subcircuits when creating net names. The character must occur between quotation marks ("). The default is “_”.

This variable replaces PEX_FMT_SPICE_NET_NAME_SEPARATOR which was too long for some versions of Linux.

## Examples

To use an underbar character as the net separator:

```
setenv PEX_FMT_SPICE_PIN_SEPARATOR "_"
```

## Related Variables and SVRF Statements

Table 14-3, “Variables for Tailoring HSPICE Netlists” on page 178

# PEX_FMT_SPICE_USE_SHORT_NET_NAMES

**Note**

As of version 2008.4, this variable is deprecated and replaced by the SHORTPINNAMES keyword in the PEX Netlist… HSPICE statement.

## Usage

PEX_FMT_SPICE_USE_SHORT_NET_NAMES {OFF | ON}

## Description

When you set this variable to ON, the formatter will replace pin names with an integer when creating SPICE net names. Not used with simple output mode.

## Related Variables and SVRF Statements

Table 14-3, “Variables for Tailoring HSPICE Netlists” on page 178

# PEX_FMT_SPICE_USE_SHORT_NAMES

**Note**

As of version 2008.4, this variable is deprecated and replaced by the SHORTPINNAMES keyword in the PEX Netlist… HSPICE statement.

## Usage

PEX_FMT_SPICE_USE_SHORT_NAMES {OFF | ON}

## Description

When you set this variable to ON, the formatter will replace pin names with an integer when creating SPICE net names. Not used with simple output mode.

This variable replaces PEX_FMT_SPICE_USE_SHORT_NET_NAMES which was too long for some versions of Linux.

## Related Variables and SVRF Statements

Table 14-3, “Variables for Tailoring HSPICE Netlists” on page 178

# PEX_FMT_SUPPRESS_DSPF_SUBCKT

**Note**

As of version 2008.4, this variable is deprecated and replaced by the NOTOPSUBCKT keyword in the PEX Netlist… DSPF statement.

## Usage

PEX_FMT_SUPPRESS_DSPF_SUBCKT

## Description

Suppresses the top level subcircuit in DSPF netlist output by the formatter. This is useful when your testbench already has the top level subcircuit and you want to include the parasitics, or if you are having trouble producing netlists simulatable by the PrimeTime® static timing analysis tool.

## Examples

To suppress the top-level subcircuit:

```
setenv PEX_FMT_SUPPRESS_DSPF_SUBCKT
```

To restore default functionality:

```
unsetenv PEX_FMT_SUPPRESS_DSPF_SUBCKT
```

## Related Variables and SVRF Statements

Table 14-5, “Variables for Tailoring DSPF Netlists” on page 179

PEX Netlist in the *Standard Verification Rule Format (SVRF) Manual.*

# PEX_FMT_UNSHORT_DEV_PIN_R

**Note**

This variable and PEX_FMT_UNSHORT_DEVICE_PIN_MODE_R are deprecated as of the 2009.1 release. Use the PEX Netlist Unshort Device Pins SVRF statement instead.

## Usage

PEX_FMT_UNSHORT_DEV_PIN_R ***R_value***

## Description

Sets the value to use for nominal resistors connecting shorted device pins. A value greater than zero creates resistors with that value. Specifically, each device pin receives its own node name, and the small-valued resistor you specify connects them.

By default, this variable is set to 0.01 ohms for CalibreView, DSPF, and SPEF formats and not set (that is, no nominal resistors are added) for other formats.

This variable replaces PEX_FMT_UNSHORT_DEVICE_PIN_MODE_R which was too long for some versions of Linux.

# PEX_FMT_UNSHORT_DEVICE_PIN_MODE_R

**Note**

This variable and PEX_FMT_UNSHORT_DEV_PIN_R are deprecated as of the 2009.1 release. Use the PEX Netlist Unshort Device Pins SVRF statement instead.

## Usage

PEX_FMT_UNSHORT_DEVICE_PIN_MODE_R ***R_value***

## Description

Sets the value to use for nominal resistors connecting shorted device pins. A value greater than zero creates resistors with that value. Specifically, each device pin receives its own node name, and the small-valued resistor you specify connects them.

By default, this variable is set to 0.01 ohms for CalibreView, DSPF, and SPEF formats and not set (that is, no nominal resistors are added) for other formats.

# PEX_GROSS_VIA_REDUC

**Note**

The PEX_GROSS_VIA_REDUC environment variable is deprecated as of the 2006.4 release. It has been replaced by the PEX Via Reduction Resistance SVRF statement.

## Usage

PEX_GROSS_VIA_REDUC {**ON** | **OFF**}

## Description

Controls whether via clusters are reduced. ON by default; set to OFF to disable reduction.

Via reduction occurs during PDB generation. If PEX_GROSS_VIA_REDUC is set to off, no reduction happens at all, and the other via reduction variables are ignored.

# PEX_HIGH_RSHEET_DEFAULT

## Usage

PEX_HIGH_RSHEET_DEFAULT ***value***

## Description

Specifies a sheet resistance value in ohms per square to use for layers that are not explicitly defined and that do not appear in device layers.

The default sheet resistance value is 0. This can result in false short circuits when a layer is in a Connect statement but does not have a specified resistance. Set this variable to a high value to detect paths created by bad rule file settings and to reduce the effects of parallel resistors.

The PEX_HIGH_RSHEET_DEFAULT value is used only for layers that do not appear in device statements. Pin and seed layers must have resistance explicitly set.

## Examples

To set the variable to 100 ohms/square in a C shell, use the following:

```
setenv PEX_HIGH_RSHEET_DEFAULT 100
```

## Related Variables and SVRF Statements

Device in the *Standard Verification Rule Format (SVRF) Manual.*

# PEX_NASSDA

**Note**

As of version 2008.4, this variable is deprecated and replaced by the HSIM keyword in the PEX Netlist… DSPF statement.

## Usage

PEX_NASSDA {**OFF** | **ON**}

## Description

Specifies to use settings for producing a netlist suitable for use with Synopsys HSIMplus Co-Simulation. The setting is only checked when the output format is DSPF.

When the PEX_NASSDA variable is set, feedthrough nets are modeled within the cell and any extracted coupling caps are listed by NET and removed from the Instance section.

## Examples

```
setenv PEX_NASSDA ON
```

# PEX_PORTS_MARK_SIGNALS

## Usage

PEX_PORTS_MARK_SIGNALS {**ON** | **OFF**}

## Description

Controls whether ports are annotated like signals. Default is to annotate (on).

Use PEX_PORTS_MARK_SIGNALS to include or exclude ports from annotation. Specify ports and xcell pin layers in Port Layer Text specification statements. You can choose to exclude ports by setting PEX_PORTS_MARK_SIGNALS to OFF.

Annotations use text from the layer in Port Layer Text. The text can be either part of the layout, or added using the Layout Text statement.

## Examples

```
setenv PEX_PORTS_MARK_SIGNALS ON
```

## Related Variables and SVRF Statements

PEX_FLOATING_NET_COUPLING

Layout Text in the *Standard Verification Rule Format (SVRF) Manual.*

Port Layer Text in the *Standard Verification Rule Format (SVRF) Manual.*

# PEX_SLOT_AREA_RATIO

**Note**

The PEX_SLOT_AREA_RATIO environment variable has been deprecated as of the 2008.3 release. It has been replaced by the PEX Slots Handling SVRF statement.

## Usage

PEX_SLOT_AREA_RATIO ***N***

## Description

The PEX_SLOT_AREA_RATIO only has an affect if PEX_SLOT_COUNT_THRESHOLD is also set. Set N to the maximum percentage that slots may occupy in the polygon. If slots account for more than that percentage of total area, the metal is NOT considered slotted. The default value is 0.3.

See "Modeling Slotted Metal" on page 128 for a description of how slotted metal is handled.

## Related Variables and SVRF Statements

PEX_SLOT_COUNT_THRESHOLD

# PEX_SLOT_COUNT_THRESHOLD

**Note**

The PEX_SLOT_COUNT_THRESHOLD environment variable has been deprecated as of the 2008.3 release. It has been replaced with the PEX Slots Handling SVRF statement.

## Usage

PEX_SLOT_COUNT_THRESHOLD ***N***

## Description

Controls whether slotted polygons are treated differently than non-slotted. Set to the minimum number of slots to distinguish between holes and slotting. Default is to treat all as non-slotted.

See “Modeling Slotted Metal” on page 128 for a description of how slotted metal is handled.

## Related Variables and SVRF Statements

PEX_SLOT_AREA_RATIO

# PEX_SLOT_SHAPE_FACTOR

**Note**

The PEX_SLOT_SHAPE_FACTOR environment variable has been deprecated as of the 2008.3 release. It has been replaced with the PEX Slots Handling SVRF statement.

## Usage

PEX_SLOT_SHAPE_FACTOR ***value***

## Description

Controls correction factor for slotted metal resistivity. The default value is 1.0. This adjustment factor is used to account for non-square slots in metal and adjusts the Rsh value used for calculating parasitic resistance.

See "Modeling Slotted Metal" on page 128 for a description of how slotted metal is handled.

## Examples

For a round metal slot use:

```
PEX_SLOT_SHAPE_FACTOR 0.9
```

## Related Variables and SVRF Statements

PEX_SLOT_AREA_RATIO

PEX_SLOT_COUNT_THRESHOLD

# PEX_TEXT_HPORTS

## Usage

PEX_TEXT_HPORTS {**OFF** | **ON**}

## Description

Uses available port text for a net in an xcell to specify the port location for that net. If no port text is available on a given net, the default mechanism for choosing the port location is used instead.

This functionality is supported only in full hierarchical mode.

# PEX_USE_ACTUAL_WIDTH

**Note**

This environment variable is deprecated as of the 2009.1 release. Use the PEX Indie Spacing SVRF statement instead.

## Usage

PEX_USE_ACTUAL_WIDTH {**ON** | **OFF**}

## Description

This variable should be set by the engineer providing the process rules. Do not change this variable

When using in-die variation, sets resistance calculations to use the actual width instead of the drawn width. Does not affect capacitance.

**Caution**

Only set this variable to ON if your variation tables are based on actual width instead of drawn width, or the calculated parasitics will be incorrect.

When including in-die variation, the Calibre xRC software uses a different resistance value based upon actual width. The resistance rules reflect in-die information provided by the foundry. This in-die information can be provided in terms of drawn width or actual width. You will need to check with the rule provider if you are not sure which type it is.

# PEX_VIA_REDUCTION_COUNT

**Note**

This environment variable is deprecated as of the 2006.4 release. It has been replaced by the PEX Via Reduction Resistance SVRF statement.

## Usage

PEX_VIA_REDUCTION_COUNT ***value***

## Description

For standard via reductions, specifies the maximum number of vias per group. Default is not set (no maximum). If set, overrides the PEX_FLEX_VIA_REDUCTION variable.

For a description of how standard via reduction works, see PEX Via Reduction Resistance in the *Standard Verification Rule Format Manual*.

## Related Variables and SVRF Statements

PEX_GROSS_VIA_REDUC

PEX_VIA_REDUCTION_LIMIT

# PEX_VIA_REDUCTION_LIMIT

**Note**

This environment variable is deprecated as of the 2006.4 release. It has been replaced by the PEX Via Reduction Resistance SVRF statement.

## Usage

PEX_VIA_REDUCTION_LIMIT ***value***

## Description

Specifies the minimum gap separating clusters of vias. Default is 5 microns.

For a description of how standard via reduction works, see PEX Via Reduction Resistance in the *Standard Verification Rule Format Manual*.

## Related Variables and SVRF Statements

PEX_GROSS_VIA_REDUC

# 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

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**

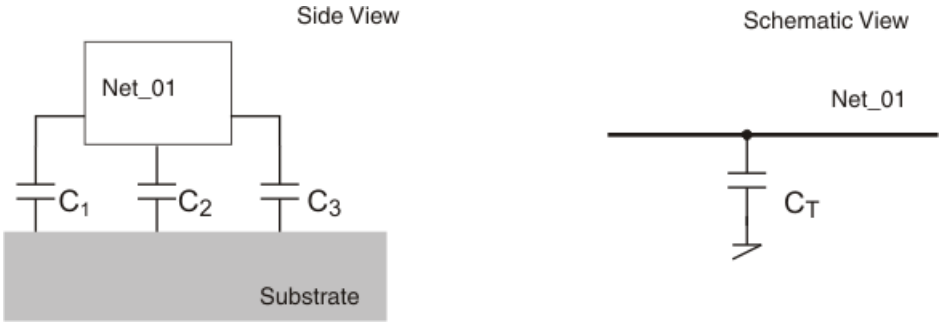


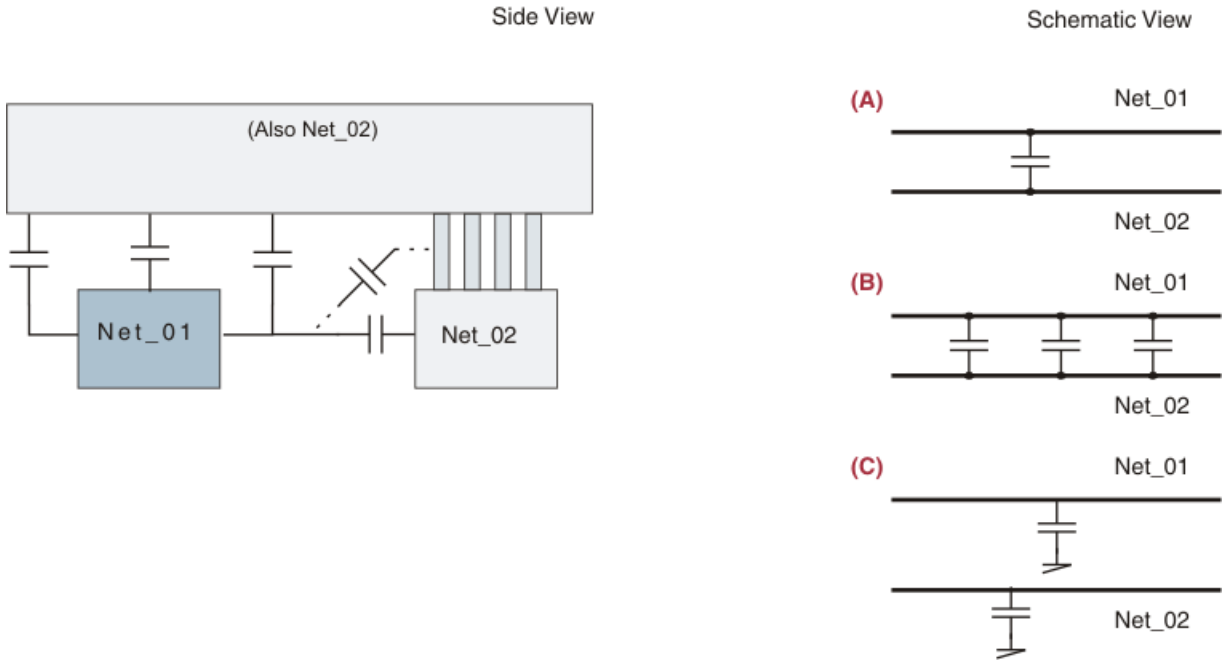

Figure A-1 shows a cross section and schematic view of intrinsic capacitance. The three parasitic capacitors shown in the layout represent the individual parasitic effects calculated by the calibrated rules. When the Calibre xRC formatter runs, the separate capacitances ($C_1$, $C_2$, and $C_3$ in the figure) are combined to a single intrinsic capacitance ($C_T$ in the figure). Intrinsic capacitance appears in the netlist as "ci_*instance*"[1].

1. Before v2007.4, intrinsic capacitance was c_instance.

# 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.)

**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, n2 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

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

**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 and Connect, Connect By, or Stamp
- Parasitic extraction: Capacitance Statements and Resistance Statements
- Intentional device recognition: Device
- PEX Report Lumped (even if you are doing only distributed netlists)

The rule summaries are listed in alphabetical order below. 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 Mentor Graphics publications:

- *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.

## 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.

**Note**

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.

**Note**

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 and using the following syntax:

**MASK SVDB DIRECTORY *filename* XRC**

where ***filename*** is a required string specifying an absolute or relative filename.

## PEX Report Lumped

The PEX Report Lumped statement generates a report of lumped C parasitic results for the extracted circuit.

**Note**

You must use the PEX Report Lumped for any parasitic operation regardless of whether it is distributed RC, lumped C, or simple extraction. For distributed RC and simple extraction, set the statement to NONE.

## 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 C

When creating lumped capacitance netlists, your SVRF rules specify the following:

- The netlist type and format: PEX Netlist Lumped (required)
- How the tool performs reduction: PEX Reduce Mincap or PEX Reduce Lumped C (optional)

The file must also include all the rules mentioned in "Required Set" on page 242.

## PEX Netlist Lumped

The PEX Netlist Lumped SVRF 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-3 shows an example statement.

**Example B-1. PEX Netlist Lumped Statement**

```
PEX NETLIST LUMPED "netlist.lumped" HSPICE 1e-6 SOURCE GROUND VSS
```

**Tip:** The PEX Netlist Lumped section of the *Standard Verification Rule Format (SVRF) Manual* explains usage of this statement in detail.

# Specific To Lumped C

There are two reduction statements applicable to lumped capacitance: PEX Reduce Mincap and PEX Reduce Lumped C. There is also a statement for creating reports, PEX Report Lumped.

**Note**

Under normal circumstances, you use *either* the PEX Reduce Lumped C or PEX Reduce Mincap statement in your rule file. Decide which statement best suits your results and omit the other statement, if necessary, from your rule file.

## PEX Reduce Mincap

The Calibre xRC tool relies on PEX Reduce Mincap SVRF statement for controlling and reducing coupling effects. Using this statement, you 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.

## PEX Reduce Lumped C

You reduce lumped C netlists using the PEX Reduce Lumped C SVRF statement in conjunction with a Calibre xRC tool invocation switch (-c) during parasitic database creation. Using the PEX Reduce Lumped C statement, you specify a constraint representing a threshold; the tool lumps lumped C values below this threshold with the intrinsic capacitances to ground.

## PEX Report Lumped

The PEX Report Lumped SVRF statement specifies the report's name and the output format. Example B-2 shows an example statement.

**Example B-2. PEX Report Lumped Statement**

```
PEX REPORT LUMPED report.lumped SOURCE
```

**Tip**: The "PEX Report Lumped" section of the *Standard Verification Rule Format (SVRF) Manual* explains usage of this statement in detail.

# Required for Distributed

When creating distributed RC netlists, your SVRF rules specify the following:

- The netlist type and format: PEX Netlist Distributed (required)
- Whether the tool performs reduction: PEX Reduce TICER, PEX Reduce Ronly, PEX Threshold, and PEX Tolerance Distributed
- Thresholds: PEX Reduce Mincap

The file must also include all the rules mentioned in "Required Set" on page 242.

## PEX Netlist Distributed

The PEX Netlist Distributed SVRF 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 Distributed SVRF statement. Example B-3 shows an example statement.

**Example B-3. PEX Netlist Distributed Statement**

```
PEX NETLIST DISTRIBUTED "netlist.dist" HSPICE 1e-6 SOURCE GROUND VSS
```

**Tip:** The "PEX Netlist Distributed" 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 Distributed statement:

- RLOCATION — reports resistor locations.
- RWIDTH — reports resistor widths.
- RLAYER — reports resistor layers.

# Specific To Distributed

There are six additional commands: one to model in-die variation, four netlist reduction commands, and one for creating ASCII reports.

- Parasitic Variation
- PEX Reduce TICER
- PEX Reduce Ronly
- PEX Threshold
- PEX Tolerance Distributed
- PEX Report Distributed

In extracting distributed netlists, Calibre xRC also applies the PEX Reduce Mincap rule if it is in the rule file.

## 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” on page 268.

## 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.

## PEX Report Distributed

The PEX Report Distributed SVRF statement specifies the report’s name and the output format. Example B-2 shows an example statement.

**Example B-4. PEX Report Distributed Statement**

```
PEX REPORT DISTRIBUTED report.distrib SOURCE
```

**Tip**: The "PEX Report Distributed" section of the *Standard Verification Rule Format (SVRF) Manual* explains usage of this statement in detail.

## PEX Threshold

The PEX Threshold statement decreases the size of netlists and databases. When you use this statement in your rule file, the Calibre xRC tool will use 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.

## PEX Tolerance Distributed

You can run bulk reduction during distributed RC or RCC parasitic extraction using the PEX Tolerance Distributed statement. This statement will perform limited reduction of these networks.

# 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 metal1           8
LAYER via              9
LAYER metal2           10


////////////////////////////////////////////////////////////
// ATTACH TEXT LAYERS
////////////////////////////////////////////////////////////

TEXT LAYER 50
ATTACH 50 metal1
ATTACH 50 metal2
LABEL ORDER metal1 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 metal1 poly nsd psd BY contact
CONNECT metal1 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

This appendix provides reference information on the various types of data which the Calibre xRC tool produces.



## Databases

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)

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.

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.

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.

# Logs

## 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
```



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 on page 259) and connections by abutment (Figure C-2 on page 259).

- 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**

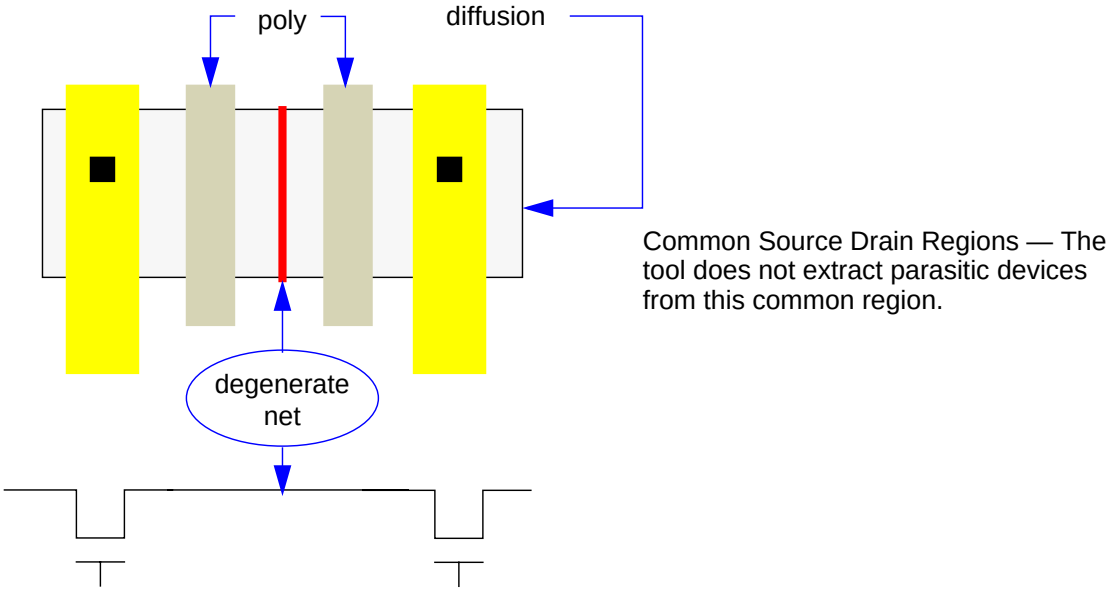
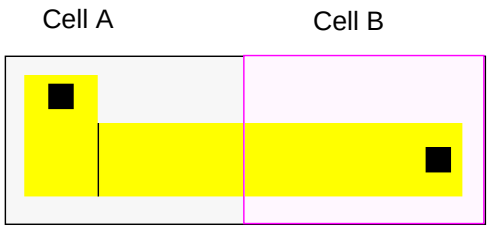



**Figure C-2. Connection by Abutment**




Connection by Abutment — During hierarchical extraction, the tool processes each cell independently. When the tool processes *Cell A*, there is no termination where the material touches the cell boundary. Therefore, the tool does not extract parasitic devices for this partial net. The same is true for *Cell B*.

# 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

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:

1. *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.
2. *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.*
3. *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.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.

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*.

## Interpreting _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 SVRF 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 environment variable PEX_FMT_NOXREF_MODEL_MODE to eliminate *_noxref* net models from the netlist.

# 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 Distributed, PEX Report Lumped, or PEX Report Netsummary SVRF 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 current working directory, and can be edited in a text editor. (Older versions of the xRC tool wrote templates in the SVDB 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 on page 265). 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 will issue 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.**

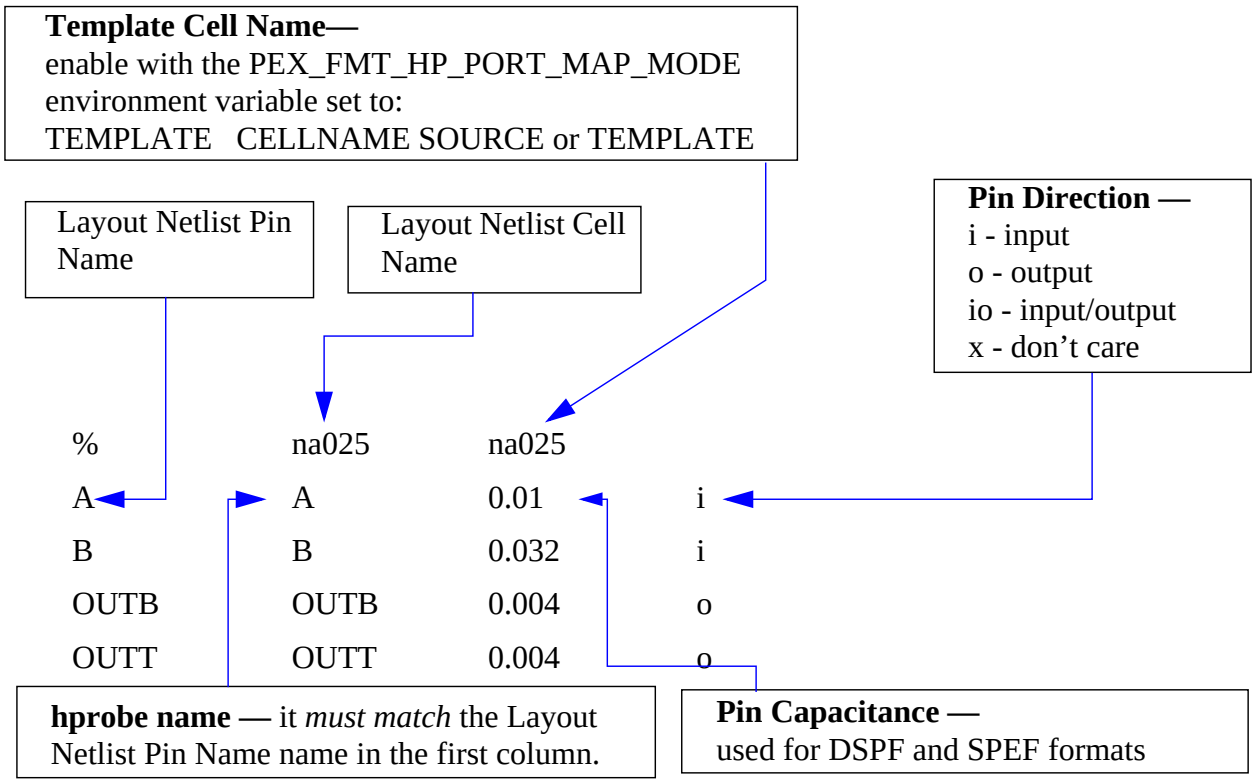
If you see this warning, the netlist is incorrect. To change port names, use the Layout Rename Text SVRF statement.

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). 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

## Capacitive And Resistive Reduction

There are four SVRF statements that reduce the number of capacitors and one SVRF statement for reducing resistors:

- PEX Reduce CC
- PEX Reduce Lumped
- PEX Reduce Mincap
- PEX Reduce Minres

PEX Reduce CC and PEX Reduce Lumped reduce 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.

**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. Mentor Graphics recommends replacing PEX Reduce Lumped with PEX Reduce CC.

### 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 Lumped

The PEX Reduce Lumped specification is applied during extraction, when the PDB is created. Coupling capacitors are compared to an absolute value. If the coupling capacitance is less than the value, it is decoupled from the nets and included in the lumped capacitance to ground.

This reduction method is no longer recommended but is retained for backwards compatibility. Because the reduction occurs during extraction it cannot be easily reversed. For more information, see "PEX Reduce Lumped" 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*.

## Threshold-based Reduction

The PEX Threshold statement decreases the size of netlists and databases. When you use this statement in your rule file, the Calibre xRC tool will use 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 delayer 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

This chapter contains the following information about the Time-it delay calculator:



## 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:

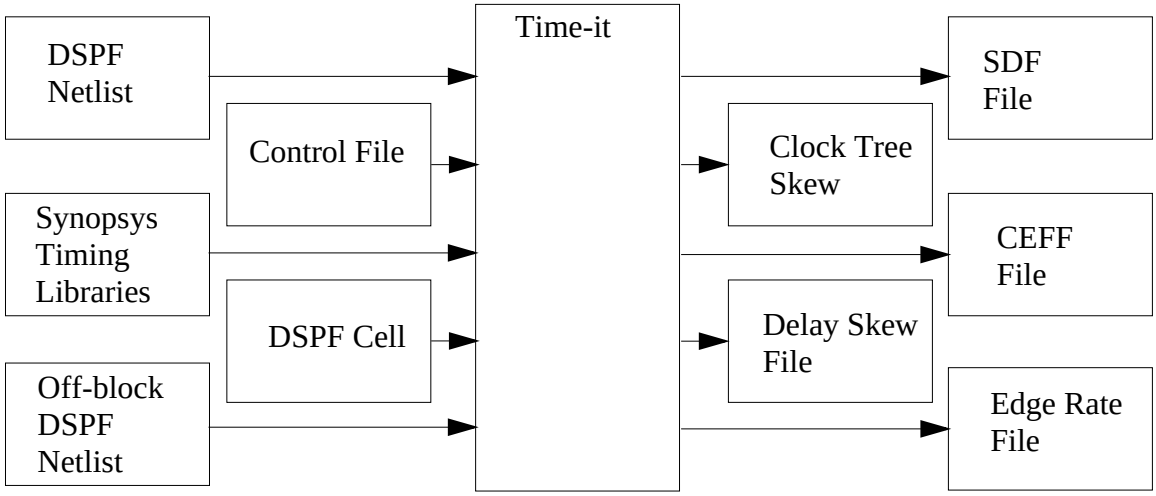
- *Time-it Reference Manual*
- *Time-it Release Notes*
- *Time-it Release Highlights*
- *Time-it Application Notes*

## 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 Synopsis 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

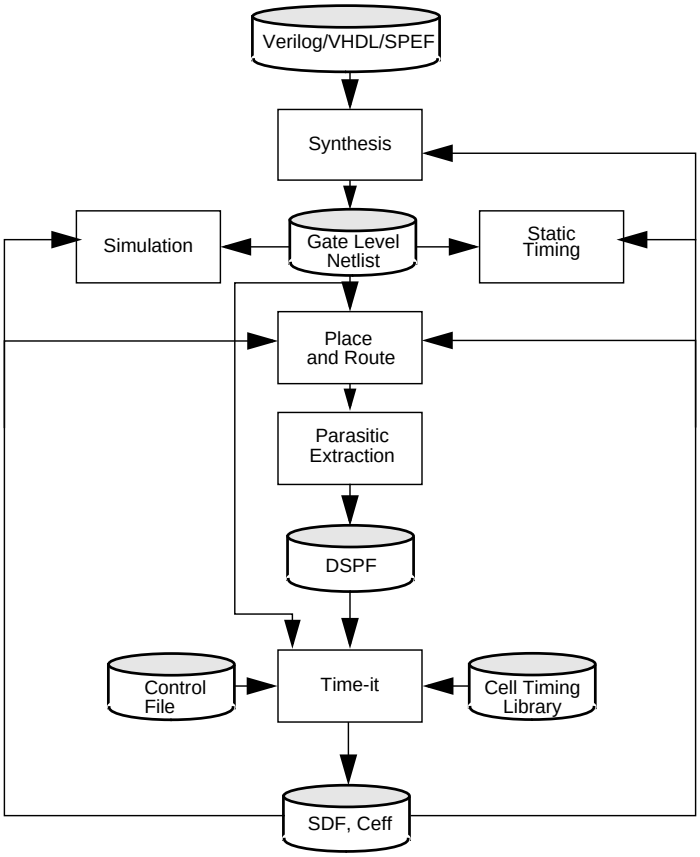
The Time-it tool outputs a Standard Delay Format (SDF) file. This file includes any cell instance pin-to-pin delays and interconnect driver-to-receiver delay, and timing check values and path constraints. A synthesis or timing analysis tool subsequently uses this netlist.

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.

# Glossary

### AMS
Analog mixed signal.

### ADvanceMS
A Mentor Graphics mixed signal design analysis tool.

### 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.

### 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 "Full Hierarchical Extraction" on page 40.

### 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" on page 46.

### hcell
A user-specified hierarchical cell used by Calibre nmLVS.

### hybrid 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*.

### 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.

### 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.

### 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.

### process corners
The variations on a “typical” process: for instance, metal thickness may not be exactly controlled.

### signal net
A grounded net that either has connections to devices or xcell ports, or is designated a port.

### 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*.

### 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.

# Index

## — W —



## — X —

# Third-Party Information

This section provides information on third-party software that may be included in the Calibre family of products, including any additional license terms.

- This software application may include Tclpro version 1.4.1 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

- This software application may include itcl/itk version 3.2.1 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

- This software application may include Tcl version 8.4.9 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

- This software application may include TkTable version 2.9 third-party software, which is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.



- This software application may include BLT version 2.4z third-party software, which is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortuous action, arising out of or in connection with the use or performance of this software.


AT&T disclaims all warranties with regard to this software, including all implied warranties of merchantability and fitness. In no event shall AT&T be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortuous action, arising out of or in connection with the use or performance of this software.


Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both copyright notice and this permission notice appear in supporting documentation, and that the names of Digital or MIT not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.
DIGITAL DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL DIGITAL BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.


IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


Silicon Metrics disclaims all warranties with regard to this software, including all implied warranties of merchantability and fitness. In no event shall Lucent Technologies be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortuous action, arising out of or in connection with the use or performance of this software.

- This software application may include Tix version 8.4.0 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.




Code derived from software contributed by the Regents of the University of California carry this disclaimer.
THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS AND CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES.

- This software application may include Milkyway version 2004.06 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

- This software application may include Tablelist version 4.1 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

  ©Csaba Nemethi All rights reserved.
  Code derived from software contributed by Csaba Nemethi are required to contain the following.

  This library is free software; you can use, modify, and redistribute it
  for any purpose, provided that existing copyright notices are retained
  in all copies and that this notice is included verbatim in any distributions.

  This software is distributed WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

- This software application may include tcllib version 1.8 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

- This software application may include portions of Boost version 1.33.1 third-party software. Boost version 1.33.1 is distributed under the terms of the Boost Software License version 1.0 and is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the license for the specific language governing rights and limitations under the license. You can view a copy of the license at: *$MGC_DOC_PATH/legal/ boost_1.0.pdf*. Boost may be subject to the following copyrights:

  ©2002-2003, Trustees of Indiana University.
  ©2000-2001, University of Notre Dame.

  Any documentation included with all redistributions must include the following acknowledgement:
  “This product includes software developed at the University of Notre Dame and the Pervasive Technology Labs at Indiana University. For technical information contact Andrew Lumsdaine at the Pervasive Technology Labs at Indiana University. For administrative and license questions contact the Advanced Research and Technology Institute at 351 West 10th Street. Indianapolis, Indiana 46202, phone 317-278-4100, fax 317-274-5902.”

  © 2001, 2002 Indiana University
  © 2000, 2001 University of Notre Dame du Lac
  © 2000 Jeremy Siek, Lie-Quan Lee, Andrew Lumsdaine
  © 1996-1999 Silicon Graphics Computer Systems, Inc.
  © 1994 Hewlett-Packard Company

  This product includes software developed at the University of Notre Dame and the Pervasive Technology Labs at Indiana University.
  Permission to use, copy, modify, distribute and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appears in all copies and that both that copyright notice and this permission notice appear in supporting documentation. Silicon Graphics makes no representations about the suitability of this software for any purpose. It is provided “as is” without express or implied warranty.

  ©2000-2005 CollabNet. All rights reserved.

  1. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

  2. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: “This product includes software developed by CollabNet (http://www.Collab.Net/).” Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

  3. The hosted project names must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact info@collab.net.

  4. Products derived from this software may not use the “Tigris” name nor may “Tigris” appear in their names without prior written permission of CollabNet.

  THIS SOFTWARE IS PROVIDED “AS IS” AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A

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

© 2000 Stephen Cleary (scleary@jerviswebb.com)

Permission to copy, use, and distribute this software and its documentation is granted, provided that the above copyright notice appears in all copies and that copyright notice appear in supporting documentation.
Permission to modify the software and its documentation, and to distribute modified software and documentation is granted, provided that: the above copyright notice appears in all copies that copyright notice appears in supporting documentation, a notice that the software was modified appears with the copyright notice.
This software and its documentation is provided “as is” without express or implied warranty, and with no claim as to its suitability for any purpose.

- This software application may include tcl threads version 2.6.3 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

- This software application may include libxml2 version 2.6.22 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. Libxml2 may be subject to the following copyrights:

  ©1991 by the Massachusetts Institute of Technology

  Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. M.I.T. makes no representations about the suitability of this software for any purpose. It is provided “as is” without express or implied warranty.

- This software application may include Combobox version 2.3 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. Combobox may be subject to the following copyright:

  © 1998-2003, Bryan Douglas Oakley
  All Rights Reserved.

  http://www.purl.org/net/oakley/tcl/combobox/index.html
  mailto:oakley@bardo.clearlight.com

  This software is provided AS-IS with no warranty expressed or implied. This software may be used free of charge, though I would appreciate it if you give credit where credit is due and mention my name when you use it.

- This software application may include Tablelist version 4.4 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

  ©Csaba Nemethi All rights reserved.
  Code derived from software contributed by Csaba Nemethi are required to contain the following.

  This library is free software; you can use, modify, and redistribute it for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions.

  This software is distributed WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

- This software application may include Bwidget version 1.7.0 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. Bwidget may be subject to the following copyrights:

  ©1998-1999 UNIFIX.
  ©2001-2002 ActiveState Corp.

  The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses.
  Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.

  IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

  THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN “AS IS” BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

- This software application may include ini third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

- This software application may include Tk version 8.4.9 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

- This software application may include TKtreectrl version 2.2.3 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

  © 1999-2000 Ajuba Solutions.
  © 2003-2005 ActiveState, a division of Sophos
  © 2002-2005 ActiveState Corporation.
  © 2002-2005 ActiveState SRL.
  © 1998-2000 by Scriptics Corporation.
  © 2002-2003 Christian Krone.
  © 2002-2006 Tim Baker

  The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.

  IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

  THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN “AS IS” BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

- This software application may include bzip2 version 1.0.1 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

- This software application may include Tcl version 8.4.9 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

- This software application may include tkcon version 2.4 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

- This software application may include clapack version 3.0 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

- This software application may include Tablelist version 4.8 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

This software is distributed WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

- This software application may include Tk version 8.4.12 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

  © 1987 by Digital Equipment Corporation, Maynard, Massachusetts, and the Massachusetts Institute of Technology, Cambridge, Massachusetts. All Rights Reserved.
  © 1990, David Koblas.
  © 1998 Hutchison Avenue Software Corporation http://www.hasc.com info@hasc.com

  © 1985, 1986, 1987, 1989, 1991 by the Massachusetts Institute of Technology
  Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the names of Digital or MIT not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

  DIGITAL DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL DIGITAL BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

- This software application may include libtecla version 1.6.1 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

  © 2000, 2001, 2002, 2003, 2004 by Martin C. Shepherd. All rights reserved.

  © 1991 by the Massachusetts Institute of Technology
  Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

  THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON- INFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

- This software application may include img version 1.2.4 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

  © 1990, David Koblas.
  Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. This software is provided “as is” without express or implied warranty.

- This software application may include Tcl version 8.5.0 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. Tcl may be subject to the following copyright:

  © 1988, 1993, 1994 The Regents of the University of California. All rights reserved.

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

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

   This product includes software developed by the University of California, Berkeley and its contributors.

4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

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

- This software application may include Tk version 8.5.0 third-party software, which is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. Tk may be subject to the following copyrights:

  © Regents of the University of California, Sun Microsystems, Inc., Scriptics Corporation, ActiveState Corporation and other parties. The following terms apply to all files associated with the software unless explicitly disclaimed in individual files.

  The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.

  IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

  THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

  © 1987 by Digital Equipment Corporation, Maynard, Massachusetts, and the Massachusetts Institute of Technology, Cambridge, Massachusetts. All Rights Reserved

  Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the names of Digital or MIT not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

  DIGITAL DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL DIGITAL BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT,

NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

© 1989, 1991 by the Massachusetts Institute of Technology

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. M.I.T. makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

- This software application may include zlib version 1.2.3 third-party software. Zlib version 1.2.3 is distributed under the terms of the zlib license and is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the license for the specific language governing rights and limitations under the license. You can view a copy of the license at: *$MGC_DOC_PATH/legal/zlib_libpng.pdf*. Zlib version 1.2.3 may be subject to the following copyrights:

  © 1995-2005 Jean-loup Gailly and Mark Adler

  This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

  2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

  3. This notice may not be removed or altered from any source distribution.

  Jean-loup Gailly jloup@gzip.org
  Mark Adler madler@alumni.caltech.edu

- This software application may include pthreads version 2.8.0 third-party software. Pthreads version 2.8.0 is distributed under the terms of the GNU Lesser General Public License version 2.1 and is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the license for the specific language governing rights and limitations under the license. You can view a copy of the license at: *$MGC_DOC_PATH/legal/gnu_lgpl_2.1.pdf*. To obtain a copy of the pthreads version 2.8.0 source code, send a request to request_sourcecode@mentor.com. This offer shall only be available for three years from the date Mentor Graphics Corporation first distributed pthreads version 2.8.0.

- This software application may include RamDebugger version 6.1.2 third-party software. RamDebugger is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

- This software application may include TclXML v3.1 third-party software, which is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. TclXML v3.1 may be subject to the following copyrights:

  © 1991 by the Massachusetts Institute of Technology

  Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. M.I.T. makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

  © Regents of the University of California, Sun Microsystems, Inc., Scriptics Corporation, and other parties. The following terms apply to all files associated with the software unless explicitly disclaimed in individual files.

The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.

IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

GOVERNMENT USE: If you are acquiring this software on behalf of the U.S. government, the Government shall have only "Restricted Rights" in the software and related documentation as defined in the Federal Acquisition Regulations (FARs) in Clause 52.227.19 (c) (2). If you are acquiring the software on behalf of the Department of Defense, the software shall be classified as "Commercial Computer Software" and the Government shall have only "Restricted Rights" as defined in Clause 252.227-7013 (c) (1) of DFARs. Notwithstanding the foregoing, the authors grant the U.S. Government and others acting in its behalf permission to use and distribute the software in accordance with the terms specified in this license.

- ]This software application may include TclDOM v3.1 third-party software, which is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

- This software application may include Expat v2.0.1 third-party software. Expat may be subject to the following copyrights:

  © 1991 by the Massachusetts Institute of Technology
  Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. M.I.T. makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

- This software application may include Poco v1.3.2 third-party software. Poco is distributed under the terms of the Boost Software License v1.0 and is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the license for the specific language governing rights and limitations under the license. You can view a copy of the license at: *$MGC_DOC_PATH/legal/boost_1.0.pdf*. Poco may be subject to the following copyrights:

  © 1995-2005 Jean-loup Gailly and Mark Adler

  This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

  2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

  3. This notice may not be removed or altered from any source distribution.

  Jean-loup Gailly Mark Adler
  jloup@gzip.org madler@alumni.caltech.edu

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly jloup@gzip.org
Mark Adler madler@alumni.caltech.edu

- This software application may include libpng v1.2.25 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. libpng v1.2.25 may be subject to the following:

  COPYRIGHT NOTICE, DISCLAIMER, and LICENSE:

  If you modify libpng you may insert additional notices immediately following this sentence.

  libpng versions 1.2.6, August 15, 2004, through 1.2.25, February 18, 2008, are Copyright (c) 2004, 2006-2008 Glenn Randers-Pehrson, and are distributed according to the same disclaimer and license as libpng-1.2.5 with the following individual added to the list of Contributing Authors:
  Cosmin Truta

  libpng versions 1.0.7, July 1, 2000, through 1.2.5, October 3, 2002, are Copyright (c) 2000-2002 Glenn Randers-Pehrson, and are distributed according to the same disclaimer and license as libpng-1.0.6 with the following individuals added to the list of Contributing Authors:
  Simon-Pierre Cadieux
  Eric S. Raymond
  Gilles Vollant
  and with the following additions to the disclaimer:

  There is no warranty against interference with your enjoyment of the library or against infringement. There is no warranty that our efforts or the library will fulfill any of your particular purposes or needs. This library is provided with all faults, and the entire risk of satisfactory quality, performance, accuracy, and effort is with the user.

  libpng versions 0.97, January 1998, through 1.0.6, March 20, 2000, are Copyright (c) 1998, 1999, 2000 Glenn Randers-Pehrson, and are distributed according to the same disclaimer and license as libpng-0.96, with the following individuals added to the list of Contributing Authors:
  Tom Lane
  Glenn Randers-Pehrson
  Willem van Schaik

  libpng versions 0.89, June 1996, through 0.96, May 1997, are Copyright (c) 1996, 1997 Andreas Dilger Distributed according to the same disclaimer and license as libpng-0.88, with the following individuals added to the list of Contributing Authors:
  John Bowler
  Kevin Bracey
  Sam Bushell
  Magnus Holmgren
  Greg Roelofs
  Tom Tanner

  libpng versions 0.5, May 1995, through 0.88, January 1996, are Copyright (c) 1995, 1996 Guy Eric Schalnat, Group 42, Inc.

For the purposes of this copyright and license, “Contributing Authors” is defined as the following set of individuals:
Andreas Dilger
Dave Martindale
Guy Eric Schalnat
Paul Schmidt
Tim Wegner

The PNG Reference Library is supplied “AS IS”. The Contributing Authors and Group 42, Inc. disclaim all warranties, expressed or implied, including, without limitation, the warranties of merchantability and of fitness for any purpose. The Contributing Authors and Group 42, Inc. assume no liability for direct, indirect, incidental, special, exemplary, or consequential damages, which may result from the use of the PNG Reference Library, even if advised of the possibility of such damage.

Permission is hereby granted to use, copy, modify, and distribute this source code, or portions hereof, for any purpose, without fee, subject to the following restrictions:

1. The origin of this source code must not be misrepresented.

2. Altered versions must be plainly marked as such and must not be misrepresented as being the original source.

3. This Copyright notice may not be removed or altered from any source or altered source distribution.

The Contributing Authors and Group 42, Inc. specifically permit, without fee, and encourage the use of this source code as a component to supporting the PNG file format in commercial products. If you use this source code in a product, acknowledgment is not required but would be appreciated.

© 1991 by the Massachusetts Institute of Technology

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. M.I.T. makes no representations about the suitability of this software for any purpose. It is provided “as is” without express or implied warranty.

© 1999 by Willem van Schaik <willem@schaik.com>

version 1.0 - 1999.10.15 - First version.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. This software is provided “as is” without express or implied warranty.

- This software application may include Bwidget version 1.8.0 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

- This software application may include TKtreectrl version 2.2.7 third-party software, which is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied TKtreectrl version 2.2.7 may be subject to the following copyrights:

  This software is copyrighted by Tim Baker and other parties. The following terms apply to all files associated with the software unless explicitly disclaimed in individual files.

  The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.

  IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS

SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THEPOSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

- ]This software application may include CLP version 1.8.2 third-party software. CLP version 1.8.2 is distributed under the terms of the Common Public License version 1.0 and is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the license for the specific language governing rights and limitations under the license. You can view a copy of the license at: *$MGC_DOC_PATH/legal/cpl_1.0.pdf*. To obtain a copy of the CLP version 1.8.2 source code, send a request to request_sourcecode@mentor.com.

- This software application may include Cbc version 2.2.1 third-party software. Cbc version 2.2.1 is distributed under the terms of the Common Public License version 1.0 and is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the license for the specific language governing rights and limitations under the license. You can view a copy of the license at: *$MGC_DOC_PATH/legal/cpl_1.0.pdf*. To obtain a copy of the Cbc version 2.2.1 source code, send a request to request_sourcecode@mentor.com.

- This software application may include Ipopt version 3.5.3 third-party software. Ipopt version 3.5.3 is distributed under the terms of the Common Public License version 1.0 and is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the license for the specific language governing rights and limitations under the license. You can view a copy of the license at: *$MGC_DOC_PATH/legal/cpl_1.0.pdf*. To obtain a copy of the Ipopt version 3.5.3 source code, send a request to request_sourcecode@mentor.com.

- This software application may include mwrap version 0.31 third-party software, which is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

- This software application may include qsort.c version 1.15 third-party software, which is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. Qsort.c version 1.15 may be subject to the following copyrights:

  © 1992, 1993 The Regents of the University of California. All rights reserved.

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

  1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

  2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

  3. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

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

- This software application may include LAPACK version 3.1.1 third-party software, which is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. LAPACK v3.1.1 may be subject to the following copyrights:

---

GAlib is available under a BSD-style license found in the COPYRIGHT file.
You can use GAlib for any purpose, and you can distribute GAlib subject to the terms in the license. If you distribute GAlib, you must include the contents of the COPYRIGHT file in your distribution.

If you use GAlib in a commercial product, you should provide credit as follows (typically in the 'about box'):

> This product includes GAlib, a library of genetic algorithm components.
> Copyright Massachusetts Institute of Technology and Matthew Wall.

To refer to GAlib from a research publication, you should provide credit as follows:

> This research was performed using GAlib, a library of genetic algorithm components (http://lancet.mit.edu/ga/).

- This software application may include Nangate Open Cell Library third-party software, which is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. Nangate Open Cell Library may be subject to the following:

  The Open Cell Library is intended for use by universities, other research activities, educational programs and Si2.org members.

  However allowed, the Open Cell Library is not intended for commercial use. If you use the Open Cell Library for demonstration of commercial EDA tools it is required to mention, indicate that the library was developed by Nangate.

  If you have questions or concerns then please contact us at openlibrary@nangate.com.

  The Open Cell Library is provided by Nangate under the following License:

  Nangate Open Cell Library License, Version 1.0. February 20, 2008

  Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the Open Cell Library and accompanying documentation (the "Library") covered by this license to use, reproduce, display, distribute, execute, and transmit the Library, and to prepare derivative works of the Library, and to permit third-parties to whom the Library is furnished to do so, all subject to the following:

  The copyright notices in the Library and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Library, in whole or in part, and all derivative works of the Library, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor. The library has been generated using a non-optimized open PDK and is not suited for any commercial purpose. Measuring or benchmarking the Library against any other library or standard cell set is prohibited. Any meaningful library benchmarking must be done in collaboration with Nangate or other providers of optimized and production-ready PDKs.

  THE LIBRARY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE LIBRARY BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE LIBRARY OR THE USE OR OTHER DEALINGS IN THE LIBRARY.

# End-User License Agreement

The latest version of the End-User License Agreement is available on-line at:
www.mentor.com/terms_conditions/enduser.cfm

**IMPORTANT INFORMATION**

**USE OF THIS SOFTWARE IS SUBJECT TO LICENSE RESTRICTIONS. CAREFULLY READ THIS LICENSE AGREEMENT BEFORE USING THE SOFTWARE. USE OF SOFTWARE INDICATES YOUR COMPLETE AND UNCONDITIONAL ACCEPTANCE OF THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT. ANY ADDITIONAL OR DIFFERENT PURCHASE ORDER TERMS AND CONDITIONS SHALL NOT APPLY.**

## END-USER LICENSE AGREEMENT ("Agreement")

**This is a legal agreement concerning the use of Software between you, the end user, as an authorized representative of the company acquiring the license, and Mentor Graphics Corporation and Mentor Graphics (Ireland) Limited acting directly or through their subsidiaries (collectively "Mentor Graphics"). Except for license agreements related to the subject matter of this license agreement which are physically signed by you and an authorized representative of Mentor Graphics, this Agreement and the applicable quotation contain the parties' entire understanding relating to the subject matter and supersede all prior or contemporaneous agreements. If you do not agree to these terms and conditions, promptly return or, if received electronically, certify destruction of Software and all accompanying items within five days after receipt of Software and receive a full refund of any license fee paid.**

1. **GRANT OF LICENSE.** The software programs, including any updates, modifications, revisions, copies, documentation and design data ("Software"), are copyrighted, trade secret and confidential information of Mentor Graphics or its licensors who maintain exclusive title to all Software and retain all rights not expressly granted by this Agreement. Mentor Graphics grants to you, subject to payment of appropriate license fees, a nontransferable, nonexclusive license to use Software solely: (a) in machine-readable, object-code form; (b) for your internal business purposes; (c) for the license term; and (d) on the computer hardware and at the site authorized by Mentor Graphics. A site is restricted to a one-half mile (800 meter) radius. Mentor Graphics' standard policies and programs, which vary depending on Software, license fees paid or services purchased, apply to the following: (a) relocation of Software; (b) use of Software, which may be limited, for example, to execution of a single session by a single user on the authorized hardware or for a restricted period of time (such limitations may be technically implemented through the use of authorization codes or similar devices); and (c) support services provided, including eligibility to receive telephone support, updates, modifications, and revisions.

2. **EMBEDDED SOFTWARE.** If you purchased a license to use embedded software development ("ESD") Software, if applicable, Mentor Graphics grants to you a nontransferable, nonexclusive license to reproduce and distribute executable files created using ESD compilers, including the ESD run-time libraries distributed with ESD C and C++ compiler Software that are linked into a composite program as an integral part of your compiled computer program, provided that you distribute these files only in conjunction with your compiled computer program. Mentor Graphics does NOT grant you any right to duplicate, incorporate or embed copies of Mentor Graphics' real-time operating systems or other embedded software products into your products or applications without first signing or otherwise agreeing to a separate agreement with Mentor Graphics for such purpose.

3. **BETA CODE.** Software may contain code for experimental testing and evaluation ("Beta Code"), which may not be used without Mentor Graphics' explicit authorization. Upon Mentor Graphics' authorization, Mentor Graphics grants to you a temporary, nontransferable, nonexclusive license for experimental use to test and evaluate the Beta Code without charge for a limited period of time specified by Mentor Graphics. This grant and your use of the Beta Code shall not be construed as marketing or offering to sell a license to the Beta Code, which Mentor Graphics may choose not to release commercially in any form. If Mentor Graphics authorizes you to use the Beta Code, you agree to evaluate and test the Beta Code under normal conditions as directed by Mentor Graphics. You will contact Mentor Graphics periodically during your use of the Beta Code to discuss any malfunctions or suggested improvements. Upon completion of your evaluation and testing, you will send to Mentor Graphics a written evaluation of the Beta Code, including its strengths, weaknesses and recommended improvements. You agree that any written evaluations and all inventions, product improvements, modifications or developments that Mentor Graphics conceived or made during or subsequent to this Agreement, including those based partly or wholly on your feedback, will be the exclusive property of Mentor Graphics. Mentor Graphics will have exclusive rights, title and interest in all such property. The provisions of this section 3 shall survive the termination or expiration of this Agreement.

4. **RESTRICTIONS ON USE.** You may copy Software only as reasonably necessary to support the authorized use. Each copy must include all notices and legends embedded in Software and affixed to its medium and container as received from Mentor Graphics. All copies shall remain the property of Mentor Graphics or its licensors. You shall maintain a record of the number and primary location of all copies of Software, including copies merged with other software, and shall make those records available to Mentor Graphics upon request. You shall not make Software available in any form to any person other than employees and on-site contractors, excluding Mentor Graphics' competitors, whose job performance requires access and who are under obligations of confidentiality. You shall take appropriate action to protect the confidentiality of Software and ensure that any person permitted access to Software does not disclose it or use it except as permitted by this Agreement. Except as otherwise permitted for purposes of interoperability as specified by applicable and mandatory local law, you shall not reverse-assemble, reverse-compile, reverse-engineer or in any way derive from Software any source code. You may not sublicense, assign or otherwise transfer Software, this Agreement or the rights under it, whether by operation of law or otherwise ("attempted transfer"), without Mentor Graphics' prior written consent and payment of Mentor Graphics' then-current applicable transfer charges. Any attempted transfer without Mentor Graphics' prior written consent shall be a material breach of this Agreement and may, at Mentor Graphics' option, result in the immediate termination of the Agreement and licenses granted under this Agreement. The terms of this Agreement, including without limitation, the licensing and assignment provisions shall be binding upon your successors in interest and assigns. The provisions of this section 4 shall survive the termination or expiration of this Agreement.

5. **LIMITED WARRANTY.**

   5.1. Mentor Graphics warrants that during the warranty period Software, when properly installed, will substantially conform to the functional specifications set forth in the applicable user manual. Mentor Graphics does not warrant that Software will meet your requirements or that operation of Software will be uninterrupted or error free. The warranty period is 90 days starting on the 15th day after delivery or upon installation, whichever first occurs. You must notify Mentor Graphics in writing of any nonconformity within the warranty period. This warranty shall not be valid if Software has been subject to misuse, unauthorized modification or improper installation. MENTOR GRAPHICS' ENTIRE LIABILITY AND YOUR EXCLUSIVE REMEDY SHALL BE, AT MENTOR GRAPHICS' OPTION, EITHER (A) REFUND OF THE PRICE PAID UPON RETURN OF SOFTWARE TO MENTOR GRAPHICS OR (B) MODIFICATION OR REPLACEMENT OF SOFTWARE THAT DOES NOT MEET THIS LIMITED WARRANTY, PROVIDED YOU HAVE OTHERWISE COMPLIED WITH THIS AGREEMENT. MENTOR GRAPHICS MAKES NO WARRANTIES WITH RESPECT TO: (A) SERVICES; (B) SOFTWARE WHICH IS LICENSED TO YOU FOR A LIMITED TERM OR LICENSED AT NO COST; OR (C) EXPERIMENTAL BETA CODE; ALL OF WHICH ARE PROVIDED "AS IS."

   5.2. THE WARRANTIES SET FORTH IN THIS SECTION 5 ARE EXCLUSIVE. NEITHER MENTOR GRAPHICS NOR ITS LICENSORS MAKE ANY OTHER WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, WITH RESPECT TO SOFTWARE OR OTHER MATERIAL PROVIDED UNDER THIS AGREEMENT. MENTOR GRAPHICS AND ITS LICENSORS SPECIFICALLY DISCLAIM ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY.

6. **LIMITATION OF LIABILITY.** EXCEPT WHERE THIS EXCLUSION OR RESTRICTION OF LIABILITY WOULD BE VOID OR INEFFECTIVE UNDER APPLICABLE LAW, IN NO EVENT SHALL MENTOR GRAPHICS OR ITS LICENSORS BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES (INCLUDING LOST PROFITS OR SAVINGS) WHETHER BASED ON CONTRACT, TORT OR ANY OTHER LEGAL THEORY, EVEN IF MENTOR GRAPHICS OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL MENTOR GRAPHICS' OR ITS LICENSORS' LIABILITY UNDER THIS AGREEMENT EXCEED THE AMOUNT PAID BY YOU FOR THE SOFTWARE OR SERVICE GIVING RISE TO THE CLAIM. IN THE CASE WHERE NO AMOUNT WAS PAID, MENTOR GRAPHICS AND ITS LICENSORS SHALL HAVE NO LIABILITY FOR ANY DAMAGES WHATSOEVER. THE PROVISIONS OF THIS SECTION 6 SHALL SURVIVE THE EXPIRATION OR TERMINATION OF THIS AGREEMENT.

7. **LIFE ENDANGERING ACTIVITIES.** NEITHER MENTOR GRAPHICS NOR ITS LICENSORS SHALL BE LIABLE FOR ANY DAMAGES RESULTING FROM OR IN CONNECTION WITH THE USE OF SOFTWARE IN ANY APPLICATION WHERE THE FAILURE OR INACCURACY OF THE SOFTWARE MIGHT RESULT IN DEATH OR PERSONAL INJURY. THE PROVISIONS OF THIS SECTION 7 SHALL SURVIVE THE EXPIRATION OR TERMINATION OF THIS AGREEMENT.

8. **INDEMNIFICATION.** YOU AGREE TO INDEMNIFY AND HOLD HARMLESS MENTOR GRAPHICS AND ITS LICENSORS FROM ANY CLAIMS, LOSS, COST, DAMAGE, EXPENSE, OR LIABILITY, INCLUDING ATTORNEYS' FEES, ARISING OUT OF OR IN CONNECTION WITH YOUR USE OF SOFTWARE AS

DESCRIBED IN SECTION 7. THE PROVISIONS OF THIS SECTION 8 SHALL SURVIVE THE EXPIRATION OR TERMINATION OF THIS AGREEMENT.

9. **INFRINGEMENT.**

9.1. Mentor Graphics will defend or settle, at its option and expense, any action brought against you alleging that Software infringes a patent or copyright or misappropriates a trade secret in the United States, Canada, Japan, or member state of the European Patent Office. Mentor Graphics will pay any costs and damages finally awarded against you that are attributable to the infringement action. You understand and agree that as conditions to Mentor Graphics' obligations under this section you must: (a) notify Mentor Graphics promptly in writing of the action; (b) provide Mentor Graphics all reasonable information and assistance to defend or settle the action; and (c) grant Mentor Graphics sole authority and control of the defense or settlement of the action.

9.2. If an infringement claim is made, Mentor Graphics may, at its option and expense: (a) replace or modify Software so that it becomes noninfringing; (b) procure for you the right to continue using Software; or (c) require the return of Software and refund to you any license fee paid, less a reasonable allowance for use.

9.3. Mentor Graphics has no liability to you if infringement is based upon: (a) the combination of Software with any product not furnished by Mentor Graphics; (b) the modification of Software other than by Mentor Graphics; (c) the use of other than a current unaltered release of Software; (d) the use of Software as part of an infringing process; (e) a product that you make, use or sell; (f) any Beta Code contained in Software; (g) any Software provided by Mentor Graphics' licensors who do not provide such indemnification to Mentor Graphics' customers; or (h) infringement by you that is deemed willful. In the case of (h) you shall reimburse Mentor Graphics for its attorney fees and other costs related to the action upon a final judgment.

9.4. THIS SECTION IS SUBJECT TO SECTION 6 ABOVE AND STATES THE ENTIRE LIABILITY OF MENTOR GRAPHICS AND ITS LICENSORS AND YOUR SOLE AND EXCLUSIVE REMEDY WITH RESPECT TO ANY ALLEGED PATENT OR COPYRIGHT INFRINGEMENT OR TRADE SECRET MISAPPROPRIATION BY ANY SOFTWARE LICENSED UNDER THIS AGREEMENT.

10. **TERM.** This Agreement remains effective until expiration or termination. This Agreement will immediately terminate upon notice if you exceed the scope of license granted or otherwise fail to comply with the provisions of Sections 1, 2, or 4. For any other material breach under this Agreement, Mentor Graphics may terminate this Agreement upon 30 days written notice if you are in material breach and fail to cure such breach within the 30 day notice period. If Software was provided for limited term use, this Agreement will automatically expire at the end of the authorized term. Upon any termination or expiration, you agree to cease all use of Software and return it to Mentor Graphics or certify deletion and destruction of Software, including all copies, to Mentor Graphics' reasonable satisfaction.

11. **EXPORT.** Software is subject to regulation by local laws and United States government agencies, which prohibit export or diversion of certain products, information about the products, and direct products of the products to certain countries and certain persons. You agree that you will not export any Software or direct product of Software in any manner without first obtaining all necessary approval from appropriate local and United States government agencies.

12. **RESTRICTED RIGHTS NOTICE.** Software was developed entirely at private expense and is commercial computer software provided with RESTRICTED RIGHTS. Use, duplication or disclosure by the U.S. Government or a U.S. Government subcontractor is subject to the restrictions set forth in the license agreement under which Software was obtained pursuant to DFARS 227.7202-3(a) or as set forth in subparagraphs (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clause at FAR 52.227-19, as applicable. Contractor/manufacturer is Mentor Graphics Corporation, 8005 SW Boeckman Road, Wilsonville, Oregon 97070-7777 USA.

13. **THIRD PARTY BENEFICIARY.** For any Software under this Agreement licensed by Mentor Graphics from Microsoft or other licensors, Microsoft or the applicable licensor is a third party beneficiary of this Agreement with the right to enforce the obligations set forth herein.

14. **AUDIT RIGHTS.** You will monitor access to, location and use of Software. With reasonable prior notice and during your normal business hours, Mentor Graphics shall have the right to review your software monitoring system and reasonably relevant records to confirm your compliance with the terms of this Agreement, an addendum to this Agreement or U.S. or other local export laws. Such review may include FLEXlm or FLEXnet report log files that you shall capture and provide at Mentor Graphics' request. Mentor Graphics shall treat as confidential information all of your information gained as a result of any request or review and shall only use or disclose such information as required by law or to enforce its rights under this Agreement or addendum to this Agreement. The provisions of this section 14 shall survive the expiration or termination of this Agreement.

15. **CONTROLLING LAW, JURISDICTION AND DISPUTE RESOLUTION.** THIS AGREEMENT SHALL BE GOVERNED BY AND CONSTRUED UNDER THE LAWS OF THE STATE OF OREGON, USA, IF YOU ARE LOCATED IN NORTH OR SOUTH AMERICA, AND THE LAWS OF IRELAND IF YOU ARE LOCATED OUTSIDE OF NORTH OR SOUTH AMERICA. All disputes arising out of or in relation to this Agreement shall be submitted to the exclusive jurisdiction of Portland, Oregon when the laws of Oregon apply, or Dublin, Ireland when the laws of Ireland apply. Notwithstanding the foregoing, all disputes in Asia (except for Japan) arising out of or in relation to this Agreement shall be resolved by arbitration in Singapore before a single arbitrator to be appointed by the Chairman of the Singapore International Arbitration Centre ("SIAC") to be conducted in the English language, in accordance with the Arbitration Rules of the SIAC in effect at the time of the dispute, which rules are deemed to be incorporated by reference in this section 15. This section shall not restrict Mentor Graphics' right to bring an action against you in the jurisdiction where your place of business is located. The United Nations Convention on Contracts for the International Sale of Goods does not apply to this Agreement.

16. **SEVERABILITY.** If any provision of this Agreement is held by a court of competent jurisdiction to be void, invalid, unenforceable or illegal, such provision shall be severed from this Agreement and the remaining provisions will remain in full force and effect.

17. **PAYMENT TERMS AND MISCELLANEOUS.** You will pay amounts invoiced, in the currency specified on the applicable invoice, within 30 days from the date of such invoice. Any past due invoices will be subject to the imposition of interest charges in the amount of one and one-half percent per month or the applicable legal rate currently in effect, whichever is lower. Some Software may contain code distributed under a third party license agreement that may provide additional rights to you. Please see the applicable Software documentation for details. This Agreement may only be modified in writing by authorized representatives of the parties. Waiver of terms or excuse of breach must be in writing and shall not constitute subsequent consent, waiver or excuse.

Rev. 060210, Part No. 227900