SIEMENS EDA

Calibre® SONR™ User's and Reference Manual

Software Version 2021.2

SIEMENS

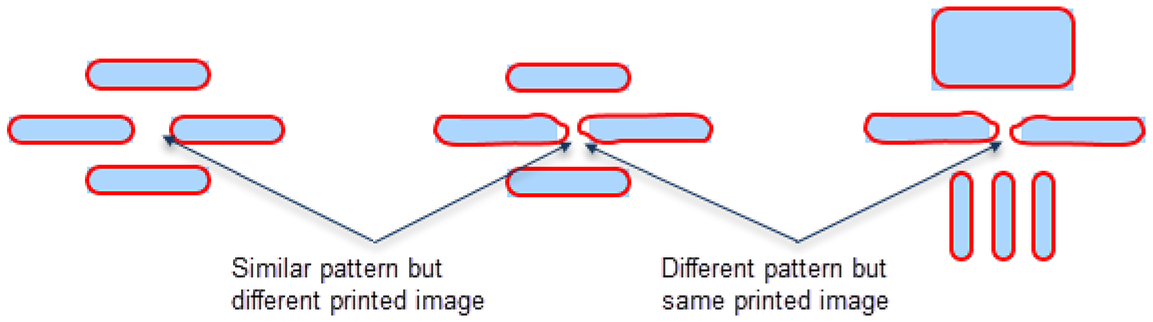
Table of Contents

List of Tables

Chapter 1
Introduction to Calibre SONR

The Calibre® SONR™ product combines multiple related machine-learning operations under a single license. The operations use feature vectors, which have been shown to correlate well with actual fabrication results. Layout shapes with similar feature vectors behave similarly in the fab.

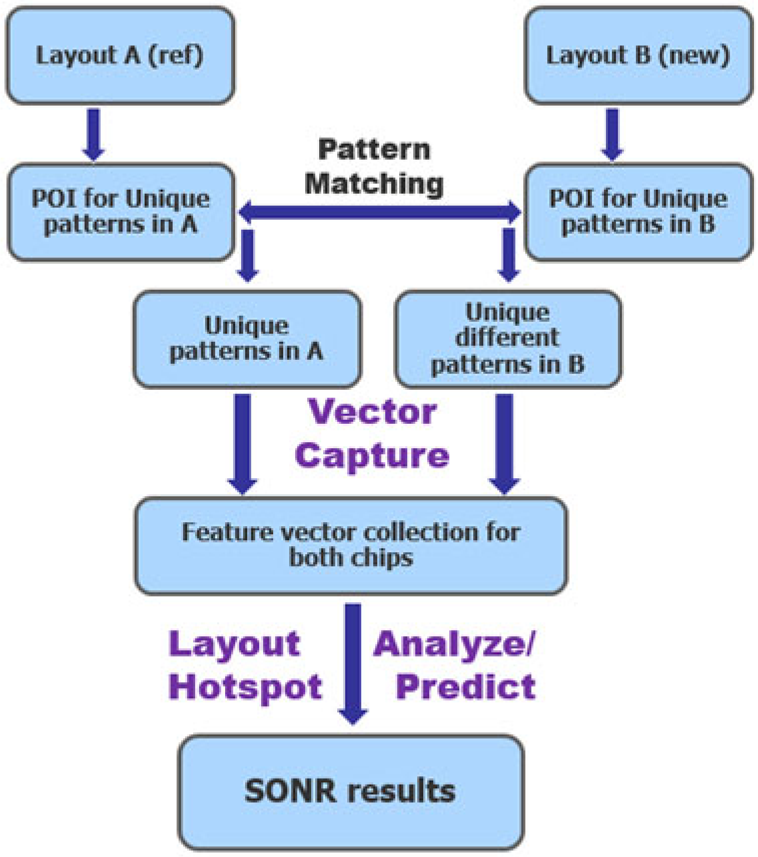
What differentiates Calibre SONR from Calibre® Pattern Matching is the ability to incorporate process information such as optical models and simulated contours. This can generate insight into why some designs yield better than others.


Similar pattern but different printed image
Different pattern but same printed image

Calibre SONR Workflow

The Calibre SONR operations can run on pre- or post-tapeout layouts. Analysis begins by creating a database representing the layout(s) and pertinent design parameters. The end result is identification of significant patterns through machine learning.

The significant patterns might be future test patterns, identified through unsupervised machine learning, or potential hotspots on new or existing layouts, identified through semi-supervised or supervised machine learning. All methods are based on the database collected by SONR Vector Capture.


Layout A (ref)
Layout B (new)
Pattern Matching
POI for Unique patterns in A
POI for Unique patterns in B
Unique patterns in A
Unique different patterns in B
Vector Capture
Feature vector collection for both chips
Layout Hotspot
Analyze/ Predict
SONR results

Layout Analyze for Test Patterns

The database contains feature vectors that describe small sections of the layout. These can either be the entire layout (provided your hardware has sufficient memory) or areas you have marked as a “point of interest” (POI). One suggested method is to use strict pattern matching to identify unique patterns. Use the SONR Vector Capture operation to create the database from the layout.

After creating a database, use SONR Layout Analyze to train a machine learning model. The model is then used in a second pass of SONR Layout Analyze to do one of the following:

- Create a unique, reduced representation of the layout, sometimes referred to as a fingerprint.
- Compare two layouts for overall similarity to give a pre-manufacturing estimate of yield.
- Identify patterns that are new to the database, and should perhaps be added to a test chip layout.

Hotspot Predict for Hotspots

The database contains feature vectors from the training data layout on regions that have been marked as hotspots or known good. The feature vectors are collected with SONR Vector Capture and SONR_COLLECT.

After creating the initial hotspot database, your Siemens representative can help you train the hotspot model. This model is then used with SONR Hotspot Predict and SONR_PREDICT on layouts to predict possible hotspots.

Calibre SONR Key Concepts

To make best use of Calibre SONR design analysis, you need to know some new types of data the software manipulates.

- **Feature vector —** A vector of up to 120 values created from measurements of a region in a layout. The feature vectors are used to “fingerprint” a layout.
- **Machine learning model —** A process-specific model created by running sets of feature vectors through the model generation and training utilities.

  Note

  Each layer requires its own model because the correlations it has encoded are tied to the process technology. Training models on data from different technologies results in spurious correlations. Applying models trained on one technology to a layer using a different technology gives incorrect results.

- **Point of interest (POI) —** A marker used to indicate an area of a layout from which to generate a feature vector.

Calibre SONR Requirements

To run Calibre SONR, you need a correctly configured environment.

Licensing

The Calibre SONR product license is required for running SVRF rule files with SONR operations. For more information on licensing, refer to the *Calibre Administrator’s Guide*.

VCO

SONR Vector Capture runs on all supported VCOs.

SONR Layout Analyze requires an AOJ Calibre tree.

Input Files

Calibre SONR works on all supported layout formats.

Syntax Conventions

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

Table 1-1. Syntax Conventions

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

Calibre SONR Modes of Operation

Calibre SONR runs in batch mode, which requires an SVRF file and invocation from a shell command line. It does not have a graphical user interface.

Calibre SONR operations are typically run with Calibre® MTflex™ multi-threading to distribute the work across hundreds of CPUs. It does not support the Calibre® FullScale™ platform. The invocation is similar to the following:

```
calibre -drc -hier -turbo rules.svrf | tee sonr.log
```

where the items in italics represent filenames that you provide, and the bold keywords are required.

Example Setup Files for Calibre SONR

Calibre SONR setup files follow the same format used in most other post-tapeout Calibre products.

SVRF File

The SVRF file identifies the layout database, maps the layers inside the layout to identifiers used by the SVRF operations, specifies settings such as database precision, and calls one or more of the Calibre SONR operations.

This example shows a simple SVRF rule file. Typical rule files include layer derivations and operations in addition to SONR VECTOR CAPTURE.

```
//Describe the input
LAYOUT PATH "input.oas"
LAYOUT SYSTEM OASIS
LAYOUT PRIMARY "*"
LAYOUT PRECISION 10000
PRECISION 10000
LAYOUT MAGNIFY AUTO

LAYER M1TARGET 1003
LAYER M1OPC    1004

//Post-tapeout run mode
LAYOUT ULTRA FLEX YES

//Describe the output
DRC MAXIMUM RESULTS ALL
DRC RESULTS DATABASE "result.oas" OASIS
DRC SUMMARY REPORT "result.report"

M1TARGET {COPY M1TARGET} DRC CHECK MAP M1TARGET OASIS 903
M1OPC    {COPY M1OPC   } DRC CHECK MAP M1OPC    OASIS 904
S_TARGET {COPY S_TARGET} DRC CHECK MAP S_TARGET OASIS 999
```

```
//Run Vector Capture
S_TARGET = SONR VECTOR CAPTURE M1TARGET M1OPC FILE sonr.setup
                                              MAP output_layer
```

Litho Setup File

The litho setup file provides settings needed by the Calibre SONR operations. It can be a separate file or placed within the SVRF file inside a LITHO FILE statement. It identifies the layers and models, and uses Tcl setlayer to create the derived layer.

This example shows the litho setup file embedded in a LITHO FILE statement, which would be inside the SVRF file.

```
LITHO FILE sonr.setup [/*
   modelpath ./
   layer target_layer
   layer marker_layer

   sonr_options SONR_OPT {
      target target_layer
      marker marker_layer
      pv_mode 1
      litho_model my_mod
      max_sampling_distance 0.08
      out_db pv.db
   }

setlayer output_layer = sonr_vector_capture target_layer marker_layer \
                                             OPTIONS SONR_OPT
*/]
```

Chapter 2
Using Calibre SONR

You must do some programming to use Calibre SONR. The information in this section describes the necessary steps to create data sets and apply models.

Collecting Model Data for Layout Analysis

In order to create a model, you must collect feature vectors that describe the layout of a layer. The feature vectors are used to train a sonr model.

Note

To collect feature vectors for interacting layers (for example, vias and the layers they connect), see "Collecting Multilayer Feature Vectors" on page 16.

Prerequisites

- An SVRF file that identifies layers in the database, precision, where to write output, and so on.
- A single layer in one or more layouts.

  If you use multiple layouts, all layers must be going through the same manufacturing steps. For example, do not collect data on both M1 and M5; nor M1 for a 22nm process and M1 from a 32nm process.

Procedure

1. (Optional.) Modify your SVRF rule file to generate a marker layer.

   The marker layer indicates where to collect feature vectors. An existing layer can also be used for a marker layer. You can use more than one marker layer.

2. Add the SONR Vector Capture command to the SVRF file.

   There are several considerations for the exact specifications you need:

   - Is the process node EUV-based? Add the EUV option. It sets the defaults to values appropriate to the smaller geometries allowed in EUV manufacturing.
   - Should the feature vectors include data generated elsewhere? If the data is attached to the layout using DFM Property, Calibre SONR can read it. Include the layers with DFM properties in the list of layers after the marker layers.

- Which layers do you want to have appear in the output layout? The Calibre engine optimizes runs; at least one layer from this operation must be written to the output through DRC Check Map. To write more than one layer out, set up parallel SONR Vector Capture statements that differ only in the MAP value.

3. Add DRC Check Map statements for the layer or layers that SONR Vector Capture maps out.

4. Create a litho setup file to configure the run.

   The litho setup file can either be separate from the SVRF file or included in it using a Litho File statement. The litho setup file should not be modified during the run.

   There is a complete example litho setup file at "Litho Setup File" on page 13. Additional commands you may want to use are listed in Table 3-1 on page 22 in Commands Grouped by Function.

   The layer name specified in the SONR Vector Capture MAP argument and after "setlayer" for sonr_vector_capture must match. Other layers are matched by order.

5. Save all files and run with calibre -drc -hier.

   A typical invocation is

   ```
   calibre -drc -hier -turbo rules.svrf
   ```

Results

When the run completes, there is a database of feature vectors as well as the usual DRC files. The database name is determined by the out_db command from Step 4. If you used the example file, the database is named *pv.db*.

Related Topics

DFM Property [Standard Verification Rule Format (SVRF) Manual]

DRC Check Map [Standard Verification Rule Format (SVRF) Manual]

Litho File [Standard Verification Rule Format (SVRF) Manual]

Collecting Multilayer Feature Vectors

In order to create a model, you must collect feature vectors that describe the layout of a layer. The set can also include interactions between layers, but the procedure is slightly different than the single-layer procedure.

Multilayer feature vectors are most often used for hotspot prediction but could also be used for other outcomes, such as comparing layouts or selecting test patterns.

Note

This procedure requires some knowledge of Calibre nmOPC tagging commands. See the *Calibre nmOPC User's and Reference Manual* for background information and tagging commands.

Prerequisites

- An SVRF file that identifies layers in the database, precision, where to write output, and so on.
- A small set of related layers, such as a via layer and the layers immediately above and below, or a set of multi-patterning layers that form one physical layer in the final chip.
- A feature vector file that includes the terms for multilayer measurements (MN and MT). You can copy one generated from another run, or contact your Siemens representative.
- A litho setup file for Calibre nmOPC. (You will not need an OPC license.)

Procedure

1. Add the SONR Vector Capture command to the SVRF file.

   There are several considerations for the exact specifications you need.

   - Is the process node EUV-based? Add the EUV option. It sets the defaults to values appropriate to the smaller geometries allowed in EUV manufacturing.
   - Should the feature vectors include data generated elsewhere? If the data is attached to the layout using DFM Property, Calibre SONR can read it. Include the layers with DFM properties in the list of layers after the marker layer(s).
   - Which layers do you want to have appear in the output layout? The Calibre engine optimizes runs; at least one layer from this operation must be written to the output through DRC Check Map. To write more than one layer out, set up parallel SONR Vector Capture statements that differ only in the MAP value.

2. Add DRC Check Map statements for the layer or layers that SONR Vector Capture maps out.

3. Create a separate litho setup file based on one used for Calibre nmOPC or Calibre nmBIAS runs for the layer that will be measured.

   Unlike single-layer vector collection, this litho setup file must be an independent file and not contained in the SVRF rule file.

   Tip

   The Calibre WORKbench RET Flow Tool, which is useful for adjusting litho setup files, preferentially shows files that end in *.in* or *.setup*. (Include the suffix in the SONR Vector Capture …FILE argument; it must exactly match the filename.)

If you have do not have access to an appropriate OPC litho setup file, ask your Siemens representative for assistance.

4. If the litho setup file does not include ml_featurevector_load, add it above the denseopc_options block.

   This command loads the feature vector file. If the file is named *phv.mod*, the command would look similar to the following:

   ```
   ml_featurevector_load nmod ./phv.mod
   ```

5. (Optional.) If you do not want to make any post-biasing or post-OPC measurements, comment out the movement sections to improve performance.

6. Create tag sets for the fragments from which measurements will be taken.

   Only layers of type opc, correct, target, sraf, and asraf can be fragmented and tagged. All layers involved in the multi-layer measurements need to be tagged.

7. Add the SONR_COLLECT command.

   Set up SONR_COLLECT for the primary layer. It is involved in all layer-to-layer measurements.

   - If there is more than one layer of type opc, or the primary layer is not of type opc, use the -layer option to identify it.
   - Use the tag set constructed of fragments from the primary layer for the -tag argument. (You can combine tag sets with "TAG or.")
   - Use the feature vector file for the -mod argument.
   - For each additional layer, supply -addLayer *layername* and optionally -mttag to restrict measurements to specific fragments on the auxiliary layers. (By default, all fragments are considered.)
   - Give a name to the output database with -outdb. If you do not do this, the collection of feature vectors is not saved.

8. For each additional layer, add a SONR_AUX_LAYER command before the SONR_COLLECT command.

   The information in SONR_AUX_LAYER must be consistent with the information in SONR_COLLECT.

   - Use -layer to identify the additional layer. It should match an -addLayer argument in SONR_COLLECT.
   - Even if you did not specify -mttag in SONR_COLLECT, you must specify a tag set in SONR_AUX_LAYER for -tag.
   - For -mod, give the name assigned to the feature vector file ("nmod" in Step 4).

- For -basemodel, specify the optical model *for this layer*. If the layer set is a via and enclosing layers, it is probably not the same optical model used for the primary layer.

9. Save both files and run.

   A typical invocation is

   ```
   calibre -drc -hier -turbo rules.svrf
   ```

Results

When the run completes, there is a database of feature vectors as well as the usual DRC files. The database name is determined by the SONR_COLLECT -outdb option.

Chapter 3
Calibre SONR Reference Information

This chapter contains descriptions for the commands used by Calibre SONR. The commands are grouped by their location in a rule file.

The SVRF rule file and litho setup file use different syntax, described in each of the “Commands” topics.

Commands Grouped by Function

The following sections group the commands according to the purpose of the run: collecting vectors, analyzing a layout, or predicting hotspots.

Collecting Vectors

Calibre SONR collects feature vectors as the raw data for machine learning models.

Table 3-1. Vector Capture Commands

Command	Description
SVRF Commands	
SONR Vector Capture	Collects data about the layout.
Setup File Commands	
setlayer denseopc and denseopc_options	For use with hotspot data and SONR_COLLECT. The setup file must be external (not in a Litho File statement) and include required dense OPC keywords.
setlayer sonr_vector_capture and sonr_options	For use with layout analysis. Collects feature vectors and writes them to a database.
Sonr_Options Block Commands	
append	Optional. Adds new feature vectors to an existing database.
dist	Optional. Specifies how far to search for a DFM property marker.
label	Optional. Adds a value to the feature vector that is not used in determining clusters.
litho_model	Optional. Identifies the location of the litho model for the process.
marker	**Required**. Identifies the layer that indicates where to collect data.
max_sampling_distance	Optional. Sets the maximum fragment length.
out_db	**Required**. Specifies a database to write the results to.
prop	Optional. Adds a value to the feature vector that is used in determining clusters.

Table 3-1. Vector Capture Commands (cont.)

Command	Description
pv_mode	**Required**. Specifies the properties to capture in the feature vectors.
target	**Required**. Identifies the layer containing layout geometry.

Analyzing a Layout

Once you have a trained machine learning model, you can analyze a layout for unique patterns that you might want to include on a test chip or to estimate wafer yield.

Table 3-2. Layout Analyze Commands

Command	Description
SVRF Commands	
SONR Layout Analyze	Invokes the layout analysis operation.
Setup File Commands	
setlayer sonr_analyze	Analyzes a layout for representative patterns.
sonr_options	Command block that contains the SONR settings.
Sonr_Options Block Commands	
append	Optional. Adds new feature vectors to an existing database.
mode	**Required**. Specifies whether the run is to train models, cluster feature vectors, or compare databases from different runs.
model	**Required**. Specifies the machine learning model to create or use.
normalize	Optional. Normalizes the input data before training.
out_db	**Required**. Specifies a database to write the results to.
poi	**Required**. Identifies the layer containing points of interest.
representative	Optional. Flags one feature vector per cluster in the database.
target	**Required**. Identifies the layer containing layout geometry.

Predicting Hotspots

Hotspot prediction requires a Calibre nmOPC-style external setup file. The SONR commands are added to this file rather than a SONR-specific one. The first step (Table 3-3) is to collect feature vectors for a hotspot model. After training the model, you can use it to predict hotspots on any layout (Table 3-4), including the one the training data was collected from.

Table 3-3. Hotspot Prediction Commands for Training Data

Command	Description
SVRF Commands	

Table 3-3. Hotspot Prediction Commands for Training Data (cont.)

Command	Description
SONR Vector Capture	Collects data about the layout.
Setup File Commands	
setlayer denseopc	Calls the denseopc_options block.
denseopc_options	Contains the SONR_COLLECT command and required dense OPC settings.
denseopc_options Block Commands	
SONR_AUX_LAYER	Optional. Identifies additional layers to use with SONR_COLLECT for multilayer feature vectors.
SONR_COLLECT	**Required**. Collects feature vectors based on the main layer and any SONR_AUX_LAYER layers.

Table 3-4. Hotspot Prediction Commands for Test Data

Command	Description
SVRF Commands	
SONR Hotspot Predict	Collects data about the layout and makes a prediction based on the hotspot model.
Setup File Commands	
setlayer denseopc	Calls the denseopc_options block.
denseopc_options	Contains the SONR_PREDICT command and required dense OPC settings.
denseopc_options Block Commands	
SONR_AUX_LAYER	Optional. Identifies additional layers to use with SONR_COLLECT for multilayer feature vectors.
SONR_PREDICT	**Required**. Analyzes the main layer and any SONR_AUX_LAYER layers and predicts hotspot locations.

Feature Vector File Format

Input for: Calibre SONR, Calibre ML OPC

The feature vector file is a type of Calibre model file. The “seed model” form (typically *phv.mod*) contains a list of features for the machine learning engine to extract. The MCOEF values are set to placeholders. The output (typically *phvo.mod*) is the same features with values for the MCOEF entries.

Format

A feature vector file must conform to the following restrictions:

- Each parameter must be on its own line.
- Each term may appear only once.
- Parameters are case-sensitive.
- Only ASCII characters are supported.
- Text after # is ignored.

This is the first four lines of a seed model feature vector file:

```
version 1
modelType NLITHO
DEFINE G1 MCOEFA 1 MCOEFB -1
DEFINE G2 MCOEFA 1 MCOEFB -1
```

This is the first four lines of an output feature vector file with real measurements for MCOEF:

```
version 1
modelType NLITHO
DEFINE G1 MCOEFA 0 MCOEFB 3
DEFINE G2 MCOEFA 1 MCOEFB 1
```

For seed models, the values for MCOEFA and MCOEFB are placeholders. When the model is generated, the values are replaced with ones calculated from the training data. A typical feature vector file has dozens, if not hundreds, of lines of DEFINE expressions.

Parameters

- **version 1**

  A required first line specifying the model version number. The only supported version is 1.

- **modelType NLITHO**

  A required second line specifying the model type. All feature vector models are NLITHO.

- **DEFINE *expression***

  A required statement that identifies a feature vector term (also referred to as a feature). The expression has the following syntax:

  ```
      term MCOEFA value MCOEFB value [TYPE kernel_expression]
  ```

  Listed below are the possible values for ***term***. Most files include at least the G and D terms.

  The groups of terms appear in approximate order of least effect on runtime to most effect. (Runtime is affected by several factors, so there is significant overlap between types of terms.)

  G1 — The direction of “outside” for the fragment.

  G2 — The corner type of the further corner of the previous fragment.

  G3 — The corner type of the further corner of the next fragment.

  G4 — The corner type of the fragment’s first corner.

  G5 — The corner type of the fragment’s second corner.

  G6 — How many fragments between the first corner and the POI (maximum 10).

  G7 — How many fragments between the second corner and the POI (maximum 10).

  G8 — The corner type of the edge in the direction of the previous fragment.

  G9 — The corner type of the edge in the direction of the next fragment.

  G10 — The corner type of the further corner of the previous edge.

  G11 — The corner type of the further corner of the next edge.

  G12 — The length of the previous fragment.

  G13 — The length of the next fragment.

  G14 — The length of the edge.

  G15 — The length of the previous edge.

  G16 — The length of the next edge.

  G17 — Distance from the POI to the internal facing edge.

  G18 — The length of the fragment that the POI is on.

  G19 — Distance from the POI to the external facing edge.

  X1 through X5 — Siemens IP.

  C1 through C32 — Siemens IP.

  CM1 through CM32 — Siemens IP.

  CP1 through CP32 — Siemens IP.

  D1 through D16 — Density terms. Density is convolved with a tophat kernel by default.

Only density terms can use the “TYPE kernel_expression” option. This can take one of two forms:

TYPE tophat *outer_radius inner_radius*

TYPE gauss *radius*

Both may have an additional MODE option (SPARSE or DENSE), which is supported for backwards compatibility but has no effect.

I1 through I12 — Image intensity terms. The image intensity assumes nominal focus and dose.

PWI1 through PWI12 — Image intensity terms, calculated with the first process window condition.

M1 through M7 — Mask terms. These terms include the effects of OPC.

RE1 through RE4 — Resist simulation terms.

SRAM — Whether the fragment interacts with SRAM.

PWRE1 through PWRE4 — Resist simulation terms calculated with the first process window condition.

R1 and R2 — Retargeting layer terms.

MN*X_term* and MT*X_term* where *X* is 1, 2, 3, or 4 and *term* is any of the other terms in this list — Multilayer terms. Calibre SONR supports up to four layers in addition to the POI layer.

Examples

The following example is sometimes referred to as “the generic 86.” It is a good starting point for a seed model.

```
version 1
modelType NLITHO
DEFINE        G1  MCOEFA   0.0 MCOEFB   3.0
DEFINE        G2  MCOEFA   0.0 MCOEFB   2.0
DEFINE        G3  MCOEFA   0.0 MCOEFB   2.0
DEFINE        G4  MCOEFA   0.0 MCOEFB   2.0
DEFINE        G5  MCOEFA   0.0 MCOEFB   2.0
DEFINE        G6  MCOEFA   0.0 MCOEFB   0.0
DEFINE        G7  MCOEFA   0.0 MCOEFB   0.0
DEFINE        G8  MCOEFA   0.0 MCOEFB   2.0
DEFINE        G9  MCOEFA   0.0 MCOEFB   2.0
DEFINE       G10  MCOEFA   0.0 MCOEFB   2.0
DEFINE       G11  MCOEFA   0.0 MCOEFB   2.0
DEFINE       G12  MCOEFA   0.0 MCOEFB   1.0
DEFINE       G13  MCOEFA   0.0 MCOEFB   2.0
DEFINE       G14  MCOEFA   0.0 MCOEFB   2.0
DEFINE       G15  MCOEFA   0.0 MCOEFB   2.0
DEFINE       G16  MCOEFA   0.0 MCOEFB   2.0
DEFINE       G17  MCOEFA   0.0 MCOEFB   2.0
DEFINE       G18  MCOEFA   0.0 MCOEFB   1.0
DEFINE       G19  MCOEFA   0.0 MCOEFB   2.0
DEFINE        X1  MCOEFA   0.0 MCOEFB   0.0
DEFINE        X2  MCOEFA   0.0 MCOEFB   0.0
DEFINE        X3  MCOEFA   0.0 MCOEFB   0.0
DEFINE        X4  MCOEFA   0.0 MCOEFB   0.0
DEFINE        X5  MCOEFA   0.0 MCOEFB   0.0
DEFINE        C1  MCOEFA  -1.0 MCOEFB   1.0
DEFINE        C2  MCOEFA  -1.0 MCOEFB   1.0
DEFINE        C3  MCOEFA  -1.0 MCOEFB   1.0
DEFINE        C4  MCOEFA  -1.0 MCOEFB   1.0
DEFINE        C5  MCOEFA  -1.0 MCOEFB   1.0
DEFINE        C6  MCOEFA  -1.0 MCOEFB   1.0
DEFINE        C7  MCOEFA  -1.0 MCOEFB   1.0
DEFINE        C8  MCOEFA  -1.0 MCOEFB   1.0
DEFINE        C9  MCOEFA  -1.0 MCOEFB   1.0
DEFINE       C10  MCOEFA  -1.0 MCOEFB   1.0
DEFINE       C11  MCOEFA  -1.0 MCOEFB   1.0
DEFINE       C12  MCOEFA  -1.0 MCOEFB   1.0
DEFINE       C13  MCOEFA  -1.0 MCOEFB   1.0
DEFINE       C14  MCOEFA  -1.0 MCOEFB   1.0
DEFINE       C15  MCOEFA  -1.0 MCOEFB   1.0
DEFINE       C16  MCOEFA  -1.0 MCOEFB   1.0
DEFINE       CP1  MCOEFA  -1.0 MCOEFB   1.0
DEFINE       CP2  MCOEFA  -1.0 MCOEFB   1.0
DEFINE       CP3  MCOEFA  -1.0 MCOEFB   1.0
DEFINE       CP4  MCOEFA  -1.0 MCOEFB   1.0
DEFINE       CP5  MCOEFA  -1.0 MCOEFB   1.0
DEFINE       CP6  MCOEFA  -1.0 MCOEFB   1.0
DEFINE       CP7  MCOEFA  -1.0 MCOEFB   1.0
DEFINE       CP8  MCOEFA  -1.0 MCOEFB   1.0
DEFINE       CP9  MCOEFA  -1.0 MCOEFB   1.0
DEFINE      CP10  MCOEFA  -1.0 MCOEFB   1.0
DEFINE      CP11  MCOEFA  -1.0 MCOEFB   1.0
DEFINE      CP12  MCOEFA  -1.0 MCOEFB   1.0
DEFINE      CP13  MCOEFA  -1.0 MCOEFB   1.0
DEFINE      CP14  MCOEFA  -1.0 MCOEFB   1.0
DEFINE      CP15  MCOEFA  -1.0 MCOEFB   1.0
DEFINE      CP16  MCOEFA  -1.0 MCOEFB   1.0
DEFINE        D1 MCOEFA   0.0 MCOEFB   1.0
DEFINE        D2 MCOEFA   0.0 MCOEFB   1.0
DEFINE        D3 MCOEFA   0.0 MCOEFB   1.0
DEFINE        D4 MCOEFA   0.0 MCOEFB   1.0
DEFINE        D5 MCOEFA   0.0 MCOEFB   1.0
DEFINE        D6 MCOEFA   0.0 MCOEFB   1.0
DEFINE        D7 MCOEFA   0.0 MCOEFB   1.0
DEFINE        D8 MCOEFA   0.0 MCOEFB   1.0
DEFINE        D9 MCOEFA   0.0 MCOEFB   1.0
DEFINE       D10 MCOEFA   0.0 MCOEFB   1.0
DEFINE       D11 MCOEFA   0.0 MCOEFB   1.0
DEFINE       D12 MCOEFA   0.0 MCOEFB   1.0
DEFINE       D13 MCOEFA   0.0 MCOEFB   1.0
DEFINE       D14 MCOEFA   0.0 MCOEFB   1.0
DEFINE       D15 MCOEFA   0.0 MCOEFB   1.0
DEFINE       D16 MCOEFA   0.0 MCOEFB   1.0
DEFINE        I1 MCOEFA   0.0 MCOEFB   1.0
DEFINE        I2 MCOEFA   0.0 MCOEFB   1.0
DEFINE        I3 MCOEFA   0.0 MCOEFB   1.0
DEFINE        I4 MCOEFA   0.0 MCOEFB   1.0
DEFINE        I5 MCOEFA   0.0 MCOEFB   1.0
DEFINE        I6 MCOEFA   0.0 MCOEFB   1.0
DEFINE        I7 MCOEFA   0.0 MCOEFB   1.0
DEFINE        I8 MCOEFA   0.0 MCOEFB   1.0
DEFINE        I9 MCOEFA   0.0 MCOEFB   1.0
DEFINE       I10 MCOEFA   0.0 MCOEFB   1.0
DEFINE       I11 MCOEFA   0.0 MCOEFB   1.0
DEFINE       I12 MCOEFA   0.0 MCOEFB   1.0
DEFINE        R1 MCOEFA   0.0 MCOEFB   1.0
DEFINE        R2 MCOEFA   0.0 MCOEFB   1.0
```

SVRF Commands

Calibre SONR is invoked using the operations below.

The SVRF rule file may use many other commands, which are described in the *Standard Verification Rule Format (SVRF) Manual*.

Table 3-5. Calibre SONR SVRF Commands

Command	Description
SONR Hotspot Predict	Analyzes a layout with a trained machine learning model and predicts hotspots.
SONR Layout Analyze	Invokes the layout analysis operation for Calibre SONR.
SONR Vector Capture	Invokes Calibre SONR to collect data about the layout.

SONR Hotspot Predict

Type: SVRF Commands

Analyzes a layout with a trained machine learning model and predicts hotspots.

Usage

SONR [EUV] **HOTSPOT PREDICT** ***target_layer***… ***marker_layer***…
  FILE ***filename*** **MAP** ***output***

Arguments

- EUV

  An optional keyword that changes the default settings to ones appropriate for extreme ultraviolet (EUV) processes. This keyword requires a Calibre EUV license.

  When EUV is specified, the litho setup file should include euv_slit_x_center.

- ***target_layer***

  A required original or derived polygon layer. Vectors are generated based on the shapes on this layer.

  You can specify multiple target layers, but there must be at least one.

- ***marker_layer***

  A required argument that specifies the name of the layer with shapes that identify points of interest (POI). The center of the bounding box of each shape is used to compute the signature.

  You can specify multiple marker layers, but there must be at least one.

- **FILE *filename***

  A required keyword specifying the path to an *external* setup file. The ***filename*** parameter can contain environment variables. For information regarding the use of environment variables in the ***filename*** parameter, refer to "Environment Variables in Pathname Parameters" in the *Standard Verification Rule Format (SVRF) Manual*.

  If you use a LITHO FILE specification statement instead of an external setup file, the run terminates with this error:

  ```
  ERROR: //-- ERROR: SONR: options block not provided.
  ```

  The setup file for SONR Hotspot Predict must contain setlayer denseopc, a denseopc_options block, and SONR_PREDICT. This command collects the vector information and predicts hotspots.

- **MAP *output***

  A required keyword that indicates the layer in the filename with data. This layer is then mapped to the layer in the SVRF rule file for output.

Description

Specifies to run Calibre SONR hotspot prediction during Calibre nmDRC-H. Hotspot prediction analyzes the layout, collects feature vectors, and applies a previously created SONR model to predict hotspots. The hotspots are written to a copy of the marker layer.

The SONR Hotspot Predict operation requires an external setup file that uses Calibre® nmOPC™ commands, including the setlayer denseopc command.

Examples

This is a complete SVRF file to run SONR Hotspot Predict. In addition to the SVRF file, the run requires an external setup file and the appropriate models.

Figure 3-1. Example SVRF File for Hotspot Prediction

```
LAYOUT SYSTEM OASIS
LAYOUT PATH "./example.oas"
LAYOUT PRIMARY "*"
DRC MAXIMUM RESULTS all
PRECISION 1000
DRC RESULTS DATABASE test_out.oas oasis pseudo

LAYER TGT    1
LAYER MARKER 100

TGT    {COPY TGT   } DRC CHECK MAP TGT    1
MARKER {COPY MARKER} DRC CHECK MAP MARKER 100
OUT_HP {COPY OUT_HP} DRC CHECK MAP OUT_HP 999

OUT_HP = SONR HOTSPOT PREDICT TGT MARKER FILE setup.in MAP predict
```

Figure 3-2. Example Setup File for Hotspot Prediction

```
## Most keywords in this file are documented in the
## Calibre nmOPC User's and Reference Manual (calbr_nmopc_useref.pdf) or
## Calibre OPCverify User's and Reference Manual (calbr_opcv_useref.pdf)
modelpath ./models
tilemicrons 15

layer target
layer marker

ml_featurevector_load nmod ./phv.mod
ml_model_load smod ./snr/32/frozen_model.pb

denseopc_options SONR_OPT {
    version 1
    layer target opc    mask_layer 0
    layer marker hidden mask_layer 0

    image lmod

    fragment_min 0.010
    fragment_max 0.030
#### In a real setup file, there is typically custom fragmentation
#### rules and tagging here.

    NEWTAG all target -out allFrags
    OUTPUT_SHAPE fragment allFrags 0.001 0.002

    SONR_PREDICT -tag allFrags -mod nmod -snr_mod smod \
      -outdb SONR_PREDICT.db
}

setlayer predict = denseopc target marker MAP target OPTIONS SONR_OPT
```

SONR Layout Analyze

Type: SVRF Commands

Invokes the layout analysis operation for Calibre SONR.

Usage

SONR LAYOUT ANALYZE *target_layer marker_layer* FILE {*filename* | *name*} MAP *output*

Arguments

- ***target_layer***

  A required argument that specifies the name of the layer containing design shapes.

- ***marker_layer***

  A required argument that specifies the name of the layer with shapes that identify points of interest (POI). Ideally, this layer is generated by SONR Vector Capture.

- **FILE {*filename* | *name*}**

  Required keyword, followed by one of the following:

  A ***filename*** indicating the path to the external setup file. The ***filename*** parameter can contain environment variables. For information regarding the use of environment variables in the ***filename*** parameter, refer to "Environment Variables in Pathname Parameters" in the *Standard Verification Rule Format (SVRF) Manual*.

  The ***name*** parameter of a LITHO FILE specification statement. The setup file is specified in the Litho File statement. This is the format used in all examples in this manual.

  The setup file for SONR Layout Analyze must contain setlayer sonr_analyze and a sonr_options block. These commands determine how the analysis is run.

- **MAP *output***

  A required keyword that indicates the layer in the filename with data. This layer is then mapped to the layer in the SVRF rule file for output.

Description

Specifies to run Calibre SONR layout analysis during Calibre nmDRC-H. Layout analysis makes measurements on the geometries on ***target_layer*** in the neighborhood of POI. These measurements are encoded as a feature vector. Feature vectors are then compared, and ones with sufficiently similar values are given the same cluster identifier. The specifics of what is measured and how comparisons are made is controlled by the setup file.

This command can be specified any number of times in your rule file. Command instances that differ only by the **MAP *output*** are run concurrently.

Examples

This is a complete SVRF file to run SONR Layout Analyze.

```
LAYOUT PATH "input.oas"
DRC RESULTS DATABASE output.oas CBLOCK STRICT OASIS PSEUDO

LAYOUT SYSTEM OASIS
LAYOUT PRIMARY "*"
DRC MAXIMUM RESULTS ALL

PRECISION 1000

LAYER target 1
LAYER poi    101

la_db = SONR LAYOUT ANALYZE target poi FILE setup.in MAP la_db

target { COPY target } DRC CHECK MAP target   1
poi    { COPY poi    } DRC CHECK MAP poi      101
la_db  { COPY la_db  } DRC CHECK MAP la_db    120
laRDB  { DFM RDB la_db "sonrA.rdb" ALL CELLS NOEMPTY CHECKNAME "la_db" }
```

The setup file *setup.in* can be either an external file or in the SVRF file within a LITHO FILE statement.

Related Topics

setlayer sonr_analyze

SONR Vector Capture

Type: SVRF Commands

Invokes Calibre SONR to collect data about the layout.

Usage

SONR [EUV] **VECTOR CAPTURE** ***target_layer*** [*marker_layer*…] [*property_layer*…]
FILE ***filename*** **MAP** ***output_layer***

Arguments

- EUV

  An optional keyword that changes the default settings to ones appropriate for extreme ultraviolet (EUV) processes. This keyword requires a Calibre EUV license.

  When EUV is specified, the litho setup file should include euv_slit_x_center.

- ***target_layer***

  A required original or derived polygon layer. Vectors are generated to describe the shapes on this layer.

- *marker_layer*

  An optional original or derived polygon layer. More than one layer can be specified. If no marker layer is provided, the target layer can be used, but may lead to excessively large files and long run times.

  The shapes on the marker layer identify target layer polygons. The polygons are analyzed and receive one or more points of interest that indicate the precise location used in collecting data for a feature vector.

- *property_layer*

  An optional derived polygon layer annotated with properties. More than one property layer can be specified.

  Properties that are referenced in the run must be numeric. They can be created with LITHO OPCVERIFY, LITHO DENSEOPC, or DFM PROPERTY. The properties must be attached to polygons.

- **FILE** ***filename***

  A required argument specifying the litho setup file. The setup file may be either in a LITHO FILE statement or an external file.

  The setup file for a Vector Capture run must include setlayer sonr_vector_capture and a sonr_options block if the output is for SONR Layout Analyze. To collect vectors for use with SONR Hotspot Predict, the setup file must be an external file, and include setlayer denseopc, SONR_COLLECT, and a denseopc_options block.

- **MAP *output_layer***

  A required argument specifying the name of a layer from the setup file. The layer information is written to a derived layer to be output by DRC CHECK MAP.

  Note

  The output layer must be written to the Calibre results database, or the SONR Vector Capture operation is ignored by the Calibre engine.

Description

This command collects feature vectors for use with other Calibre SONR operations. A feature vector is generated for the environment around each point of interest. The information captured by the feature vector depends on the models and other settings in the setup file.

Although the primary output is a database of feature vectors, SONR VECTOR CAPTURE creates a derived layer, which must be written to a results database to prevent the compiler optimizing away the operation.

The name of the database of feature vectors is set in the litho setup file by the out_db command.

Examples

Example 1 - Writing Output Layer

The following examples show SONR VECTOR CAPTURE within the context of an SVRF section. Notice how the dummy output layer is written in a DRC CHECK MAP statement to keep the Calibre optimizer from eliminating the operation.

```
DUMMY_OUT = SONR VECTOR CAPTURE v12 FILE svc.in MAP DUMMY_OUT
DUMMY { COPY DUMMY_OUT } DRC CHECK MAP DUMMY
```

This could also be written as follows:

```
DUMMY { SONR VECTOR CAPTURE v12 FILE svc.in MAP DUMMY_OUT }
DRC CHECK MAP DUMMY
```

Example 2 - Determining Point of Interest

In the simplest case, a marker layer containing small (1 dbu) squares is passed. When there is no marker layer, the shapes on the target layer are used. Edges are split into shorter pieces, and each piece receives a marker in the center. This can result in a huge number of feature vectors, most of which are redundant.

One way to create a marker layer is to use other SVRF commands such as INTERNAL or WITH WIDTH to select particular polygons of interest:

```
//Identify corners and expand the shapes on the edge layer into polygons
convex_corner = INTERNAL m1 <= 0.050 ABUT ==90 INTERSECTING ONLY
concave_corner = EXTERNAL m1 <= 0.05 ABUT ==90 INTERSECTING ONLY
all_corners = OR EDGE convex_corner concave_corner
corner_markers = EXPAND EDGE all_corners BY 0.001
```

You would, of course, want to use more precise selection criteria than the preceding code. It demonstrates the concept, but returns even more points of interest than the default method.

If you have run OPC on the layout, another method to identify points of interest is to use LITHO OPCVERIFY checks:

```
   bridges = LITHO OPCVERIFY m1.out sraf m1.orig FILE m1.orc MAP bridge_check
```

Related Topics

setlayer sonr_vector_capture

euv_slit_x_center [Calibre OPCverify User's and Reference Manual]

Model Generation Commands

To train a machine learning model based on SONR Vector Capture output, use the sonr command line utility. It runs in a Linux shell rather than a Calibre rule file.

The sonr command line utility is case-sensitive. To see a complete list of options, trigger the usage message:

```
$MGC_HOME/bin/sonr
```

Table 3-6. sonr Application Commands

Command	Description
sonr --cluster	Creates clusters of feature vectors, which can be used to select representative features or generate a sonr model.
sonr --concatenate_cv	Combines two CSV files or two databases.
sonr --model_creator	Creates machine learning models.
sonr --read	Writes feature vector terms from a file pool into a CSV or HDF5 file.
sonr --tree	Creates a prediction tree using a previously trained cluster model.

sonr --cluster

Type: Model Generation Commands

Creates clusters of feature vectors, which can be used to select representative features or generate a sonr model.

Usage

sonr --cluster {**-i** | **--in**} ***data_file***… [-h | --help]
[*Input Options*] [*Clustering Options*] [*Output Options*]

Input Options:
[--cs *col*] [--ce *col*] [--phv *file*] [--load *model*]

Clustering Options:
[-n *int*]
{[--target_clusters *int*] | [--target_fa *float* [--hsl *col*] [--hsv *int*…]] }
[--n_tolerance *int*] [--norm {<u>0</u> | 1}]
[--dsel {0 | <u>1</u>}] [--ddsel {<u>0</u> | 1 | 2}]

Output Options:
[--keep {<u>0</u> | 1}] [{-o | --out} *data_file*] [--save *directory*]

Arguments

- {**-i** | **--in**} ***data_file***

  A required argument that specifies the input data to use for training the model or making a prediction. The supported formats are CSV, HDF5, and file pool. The SONR applications produce file pool output.

  To use CSV, the data file should have a *.csv* suffix. To use HDF5 format, the data file should have an *.hdf5* suffix. File pool data files may end in *.db* or *.fp*. If the data file does not have a recognized suffix, it is treated as a file pool file.

  To supply more than one file, separate each filename with a space. The files are concatenated, and only features common to all files are retained.

- -h | --help

  An optional argument that prints an extended usage message to the terminal and exits. All other arguments are ignored when -h or --help is specified.

Input Options

- --cs *col*

  An optional argument that specifies the heading of the column to use as the start of feature input. The default value is G1.

  This argument is ignored if --phv is specified.

- --ce *col*

  An optional argument that specifies the heading of the column to use that is the end of the feature input. The default value is I12.

  This argument is ignored if --phv is specified.

- --phv *file*

  An optional argument that loads a file to determine feature input. The file is typically named *phv.mod*. Ask your Siemens representative for a “seed model” reflecting the most recent best practices, or copy the example in “Feature Vector File Format” on page 25.

  The --phv argument takes precedence over --cs and --ce.

- --load *model*

  An optional argument that specifies the name of a directory containing a clustering model.

  Loading a model triggers a prediction run; the input data does not contribute to the clusters.

Clustering Options

- -n *int*

  An optional argument that specifies the resolution to use for calculating the number of potential clusters. Larger values produce more clusters. The default value is 16.

- --target_clusters *int*

  An optional argument that specifies how many clusters you would like produced from the training data, and from that calculates a resolution value (-n). If specified with -n, the specified resolution is used as the initial value when searching for the resolution that achieves the desired number of clusters.

  The --target_clusters option is not guaranteed to produce the best resolution value, nor to deliver exactly *int* clusters. It cannot be specified with --target_fa.

- --target_fa *float* [--hsl *col*] [--hsv *int…*]

  An optional argument set that specifies the false alarm rate and from that calculates a resolution value (-n). The rate specified by *float* must be between 0 and 1.

  If both --target_fa and -n are specified, the value of -n is used as the initial value when searching for the resolution that achieves the desired false alarm rate.

  The --target_fa option is not guaranteed to produce the best resolution value, nor to find a value that achieves the target false alarm rate. It cannot be specified with --target_clusters.

  The --target_fa argument can specify the data to use for hotspots with the following options:

  --hsl *col* — Specifies the column label of the column with the hotspot data. The default is Reta. Label names are case-sensitive.

  --hsv *int…* — Specifies the value in the hotspot column that identifies a hotspot. You can specify multiple values by separating them with commas. The default value is -1.

- --n_tolerance *int*

  An optional argument that is used in conjunction with --target_clusters or --target_fa. The --n_tolerance value sets the number of trials for the calculation. Larger values of *int* produce more optimal results, but take longer. The default value is 10.

- --norm {0 | 1}

  An optional argument that specifies whether to normalize data prior to clustering. The default is no normalization (0).

  The normalization state is saved within the model. This ensures that prediction runs also normalize input to match the model, if needed.

- --dsel {0 | 1}

  An optional argument that "down selects" a cluster. Use --dsel 1 to output a representative feature vector per cluster. Use --dsel 0 to improve performance for hotspot prediction.

- --ddsel {0 | 1 | 2}

  An optional argument that specifies a method of choosing a representative feature vector from a cluster.

  0 — Choose a feature vector per cluster randomly. This is the default behavior.

  1 — Use the feature vector closest to the centroid of the cluster.

  2 — Use the feature vector closest to the center of the cluster based on the boundaries of the grid.

Output Options

- --keep {0 | 1}

  An optional argument that specifies whether the output should contain only the representative feature vectors (0) or all feature vectors (1). The default behavior is to keep only the representative feature vectors.

- {-o | --out} *data_file*

  An optional argument that specifies where to write model prediction results. The filename must end in *.fp* or *.db*.

  If this argument is not specified, sonr --cluster does not write out results.

- --save *directory*

  An optional argument that specifies a directory to save the cluster model to.

  Caution

  Do not use an existing directory. The directory is emptied before the model is written to it.

Description

The optional sonr --cluster application produces a cluster model. In cluster models, the feature vectors are grouped by similarity.

Cluster models can be used in the following ways:

- As an initial model for sonr --tree. Provide the directory specified in --save to the --im argument in the first sonr --tree pass.
- To select representative feature vectors. Representative feature vectors are useful for comparing layouts for yield predictions and selecting test patterns to ensure sufficient diversity on test wafers.

Related Topics

sonr --tree

SONR_PREDICT

sonr --concatenate_cv

Type: Model Generation Commands

Combines two CSV files or two databases.

Usage

sonr --concatenate_cv --csvf1 *file1* --csvf2 *file2* [--phvo1 *mod1* --phvo2 *mod2*] [--csvout *file*] [--phvout *mod*]

Arguments

- **--csvf1 *file1* --csvf2 *file2***

  A required pair of arguments specifying the two files to combine. The files must be the same type, and must capture the same measurements. (That is, the seed model used in their generation must have used the same set of terms.)

  Only two files can be specified. They must both be CSV format or both be MLDB database.

- --phvo1 *mod1* --phvo2 *mod2*

  An optional argument specifying the models associated with the two files. This argument is required when ***file1*** and ***file2*** are CSV format.

  The models are feature vectors files, with calibrated MCOEFA and MCOEFB values.

- --csvout *file*

  An optional argument naming the CSV output file. This can only be used with CSV input and if not used, gives the error "'NoneType' object is not callable."

  Attempts to create a CSV output from MLDB databases generates the error "Cannot create a non-filepool output with filepool inputs!"

- --phvout *mod*

  An optional argument naming the model generated from the combined files. The default name is *phvo_cat.mod*.

Description

The optional sonr --concatenate_cv application combines two sets of feature vectors provided that both sets were produced with the same seed models.

Examples

To combine MLDB databases, only the required arguments are needed:

```
sonr --concatenate_cv --csvf1 database1.fp --csvf2 database2.fp
```

The combined databases are saved in the file *calibration_vectors_cat.db*.

To combine CSV files, use this syntax:

```
sonr --concatenate_cv --csvf1 file1.csv --csvf2 file2.csv \
                      --phvo1 file1.mod --phvo2 file2.mod \
                      --csvout file3.csv
```

sonr --model_creator

Type: Model Generation Commands

Creates machine learning models.

Usage

sonr --model_creator model_2 [-h] {**--i** ***data_file***}… [--o *directory*] **--f** ***seed_model***
--l ***label*** [--p *number*] [--n *number*]

Arguments

- -h

  An optional argument that prints an extended usage message to the terminal and exits. All other arguments are ignored when -h is specified.

- **--i** ***data_file***

  A required argument that specifies the input data to use for creating a model. The supported formats are CSV and file pool.

  The data file must contain labeled feature vectors to create a fully supervised (model_2 type) machine learning model. Create labeled data using either setlayer sonr_vector_capture and label or setlayer denseopc and SONR_COLLECT -record.

  You can provide more than one data file using an asterisk to match any substring or by repeating the argument. For example:

  ```
  sonr --model_creator model_2 -i set1.csv -i set2.csv …

  sonr --model_creator model_2 -i set*.csv …
  ```

- --o *directory*

  An optional argument that specifies the directory to write the model to. Use an empty or new directory. The default directory is *./model*.

  Note
  Do not use the current working directory. Any files in the output directory are deleted before the model is written.

- **--f** ***seed_model***

  A required argument that specifies a feature vector file that contains the terms to include in the model.

- **--l** ***label***

  A required argument that specifies the name of the column containing the labeled data. Label names are case-sensitive.

- --p *number*

  An optional argument that specifies the value in the label column that identifies hotspot data. The number can be a floating point number or integer. The default is 1.

- --n *number*

  An optional argument that specifies the value of the label column that identifies non-hotspot ("good") data. The number can be a floating point number or integer. The default is 0.

Description

The optional sonr --model_creator application produces a supervised machine learning model. Supervised machine learning models are created by using fully labeled data sets in the training.

The application splits the input data to use some for training and the rest for verifying its model. Because the process is fully automated, it is important to provide sufficient input data -- at least five feature vectors per feature. (For example, if the seed model has 40 terms, there should be at least 200 unique feature vectors.) Sparse training data does not provide enough examples to completely train models.

Troubleshooting

A successful run ends with a status message like the following:

```
Creating a model_2
[status]       The created model is saved to './model'.       <timestamp>
[logging] Elapsed time for step 0:           00:00:02.05
[logging] Total elapsed time:                   00:00:24.13
```

If you get an error "Encrypted file is corrupted," the most likely cause is that one or more of the arguments used only a single hyphen. All arguments except -h have two hyphens.

If the run ends with "KeyError: "['***label***'] not in index," verify the spelling of the label column. Label names are case-sensitive.

If the transcript includes the message "ValueError: Found 1 unique classes in dataFrameTrain; only 2 class SONR Supervised Type 1 models may be trained currently," verify your database includes both good and bad locations. Also check that --n and --p, if specified, are set to different values from each other.

Examples

This creates a fully supervised machine learning model from a file pool database created by SONR_COLLECT:

```
sonr --model_creator model_2 --i sonr_collect.db --f sonr_collect.mod \
                             --l pinch
```

sonr --read

Type: Model Generation Commands

Writes feature vector terms from a file pool into a CSV or HDF5 file.

Usage

sonr --read {**-i** | **--input**} ***fp*** [{-c | --columns} *col* [*col*…]] [{-o | --output} *file*]

Arguments

- {**-i** | **--input**} ***fp***

  A required argument that specifies the name of the file pool to read. File pools typically have a suffix of *.fp* or *.db*.

- {-c | --columns} *col* [*col*…]

  An optional argument that specifies specific columns to write out. The column names should be separated by spaces.

  If this argument is not present, the output includes all user-visible terms. The names of the columns are on the first row. Some terms are proprietary and are not listed.

- {-o | --output} *file*

  An optional argument that specifies a name for the output file. If *file* does not end in *.csv*, the output is in HDF5 format.

  If this argument is not present, the output is written to *dumpedFilepool.csv*.

Description

The optional sonr --read application is a utility for examining file pool content. It writes the user-visible terms of the feature vectors to a CSV or HDF5 file.

Even a simple feature vector has dozens of terms. If the file pool contains data from a full chip layout, the output file may be over a GB in size. Because of this, when doing an initial output for column names, use a smaller file pool such as one that has been down selected.

Examples

Example 1

This example returns the column headings from a file pool named *training.db* in a file named *dumpedFilepool.csv*:

```
sonr --read --input training.db
```

Example 2

This example reads three columns from *training.db* and writes them to an HDF5 format file, *representative_fragments*.

```
sonr --read -i training.db -c SELECTED X Y -o representative_fragments
```

sonr --tree

Type: Model Generation Commands

Creates a prediction tree using a previously trained cluster model.

Usage

sonr --tree --im *model* {**-i** | **--in**} ***data_file***… [-a | --append] [-h | --help] [--keep {0 | 1}]
[{-m | --mode} {0 | 1}] [--save *directory*]

Arguments

- **--im *model***

  A required argument that specifies the directory of the input model to be used for training or prediction.

  For training a prediction tree, the model must have a trained cluster model or a prediction tree model that has been through at least one round of training. For prediction, you must have a trained prediction tree model.

- {**-i** | **--in**} ***data_file***

  A required argument that specifies the input data to use for training the model or predicting hotspots. The supported formats are CSV, HDF5, and file pool. The SONR applications produce file pool output.

  To use CSV, the data file should have a *.csv* suffix. To use HDF5 format, the data file should have an *.hdf5* suffix. File pool data files may end in *.db* or *.fp*. If the data file does not have a recognized suffix, it is treated as a file pool file.

  To supply more than one file, separate each filename with a space.

- -a | --append

  An optional argument that adds entries to an existing model during training (--mode 0). The argument is ignored for --mode 1.

  When this argument is not specified, the existing model is discarded and a new prediction tree created.

- -h | --help

  An optional argument that prints an extended usage message to the terminal and exits. All other arguments are ignored when -h or --help is specified.

- --keep {0 | 1}

  An optional argument that specifies the feature vectors to output.

  0 — Output only feature vectors that were able to be predicted. This is the default.

  1 — Output all feature vectors, including those whose clusters were not found in the tree.

- {-m | --mode} {0 | 1}

  An optional argument that specifies whether to train a model or make a prediction based on the input.

  0 — Generate (train) a model. This is the default when the argument is not specified.

  1 — Make a prediction regarding the input data using the input model. Predictions are written to a binary file *pred_out.db*.

- --save *directory*

  An optional argument that specifies a directory to save the tree model to.

  Caution

  Do not use an existing directory. The directory is emptied before the model is written to it.

Description

The optional sonr --tree application creates a prediction tree from a previously trained clustering model. Each unique cluster is saved as a node within the tree.

The tree model supports incremental training, and may be used later for prediction on new data.

Examples

Example 1

This example shows a training run:

```
sonr --tree -i Train.30.fp -im sonr_tree_model4 --save sonr_tree_model5
```

The input data is *Train.30.fp*, a file pool containing previously collected vectors. The initial model is in the directory *sonr_tree_model4*. The updated model, which may have completely different clusters than *sonr_tree_model4*, is written to a new directory, *sonr_tree_model5*.

Example 2

This example shows a prediction run:

```
sonr --tree -i CustomCPD.fp -im sonr_tree_model6 --mode 1 --keep 1
```

The input data is *CustomCPD.fp*, which was created by a separate SONR Vector Capture run. It applies the prediction tree saved in the *sonr_tree_model6* directory. Because --mode 1 is present, the prediction tree is not modified. Results, including feature vectors that did not match any clusters, are written to *pred_out.db* in the current working directory.

Litho Setup File Commands

The “FILE name” argument to SONR LAYOUT ANALYZE and SONR VECTOR CAPTURE specifies a litho setup file containing a command initialization block. Commands in this block are lowercase only, and must be specified one per line.

Litho setup files can be separate files in the same directory as the SVRF rule file, or can be included in the SVRF rule file using a LITHO FILE statement.

Table 3-7. SONR Litho Setup File Commands

Command	Description
density_image_load	Loads a PNG file with precomputed density convolution.
ml_model_load	Loads the sonr model for use by SONR_PREDICT.
setlayer sonr_analyze	Analyzes a layout for representative patterns.
setlayer sonr_vector_capture	Collects feature vectors and writes them to a database.
sonr_options	Command block within a setup file that defines the Calibre SONR settings.
SONR_AUX_LAYER	denseopc only. Defines auxiliary layers for SONR_COLLECT and SONR_PREDICT.
SONR_COLLECT	denseopc only. Collects features vectors and writes them to a database for creating a hotspot prediction model.
SONR_PREDICT	denseopc only. Analyzes a layout and predicts hotspots based on trained models.

density_image_load

Type: Litho Setup File Commands

Loads a PNG file with precomputed density convolution.

Usage

density_image_load *name filepath* [no_precheck]

Arguments

- ***name***

  A required argument that specifies the name to use for the image in other commands.

- ***filepath***

  A required argument that specifies the PNG file to load. If the file is not in the current working directory or the modelpath, ***filepath*** must be a full path starting at root (/).

- no_precheck

  An optional keyword that specifies the PNG file is not checked until the operation that uses it runs. This can be useful when the same SVRF file generates the flare map and runs SONR. By default, the PNG file is verified when parsing the SVRF file.

Description

The optional density_image_load command associates a PNG file with a name. The PNG file should contain precomputed density information from Density Convolve or RET Flare_Convolve.

Density PNG files are used in the -density options of SONR_COLLECT and SONR_PREDICT. Each PNG file should have its own density_image_load instance.

Examples

Example 1: Matching Names

The following lines show how to refer to the PNG file in SONR_COLLECT:

```
density_image_load p100 ./z10_100um.png
...
SONR_COLLECT -tag allFrags -mod nmod -density p100 -outdb png.db
```

Example 2: Multiple PNG Files

The following example shows how to specify multiple PNG files.

```
density_image_load p1 ./color.png
density_image_load p2 ./grey.png
...
SONR_COLLECT -tag allFrags -mod nmod -density p1 -density p2 \
   -outdb two_png.db
```

Related Topics

Density Convolve [Standard Verification Rule Format (SVRF) Manual]

RET FLARE_CONVOLVE [Calibre nmOPC User's and Reference Manual]

ml_model_load

Type: External setup file command

Loads the sonr model for use by SONR_PREDICT.

Usage

ml_model_load *name filepath*

Arguments

- ***name***

  A required argument specifying the name to use when referring to this model.

- ***filepath***

  A required argument specifying the filename or relative path of the sonr model. The model can be in the current directory or the modelpath.

  If no filename is given, ml_model_load searches for a file named *frozen_model.pb*. At a minimum, the command must specify the directory.

Description

The ml_model_load command loads the sonr machine learning model. It is required when using SONR_PREDICT. The file path should end in a model created by sonr --tree.

Metadata for the model is stored separately and is loaded by the ml_featurevector_load command.

This command can appear more than once in the setup file, but each instance must use a different name.

setlayer sonr_analyze

Type: Litho Setup File Commands

Analyzes a layout for representative patterns.

Usage

setlayer *out_layer* = sonr_analyze *target_layer poi_layer* options *so_opt*

Arguments

- ***out_layer***

  A required argument that specifies a unique layer name. This new layer receives the results. At a minimum this is a copy of the ***marker_layer***, and may have attached properties depending on the mode specified in the options block.

- ***target_layer***

  A required argument that specifies the name of the layer with shapes to analyze.

- ***poi_layer***

  A required argument that specifies the name of a layer with POI markers. This layer should be generated by SONR Vector Capture.

  For the cluster and clusterThenDiff modes, the ***poi_layer*** receives properties and is the basis of ***out_layer***.

- **options *so_opt***

  A required keyword and argument indicating the name of the sonr_options block to use.

Description

This command analyzes the input target layer and determines similar areas, or clusters. The meaning of "similar" depends on the settings in the sonr_options block. Use this command both to create the machine learning model and to analyze layout patterns.

Cluster identifiers are returned as properties in the output layer.

The options block must contain the following commands:

- target to identify the layout
- poi to identify which locations to analyze
- mode to specify which database to use and what to do with the data
- out_db to name the database of results.

Examples

```
setlayer sa_out = sonr_analyze target vc_out options analyze.in
```

Related Topics

SONR Layout Analyze

sonr_options

setlayer sonr_vector_capture

Type: Litho Setup File Commands

Collects feature vectors and writes them to a database.

Usage

setlayer *out_layer* = sonr_vector_capture *target_layer marker_layer* options *so_opt*

Arguments

- ***out_layer***

  A required argument that specifies a unique layer name. This new layer receives the results. At a minimum this is a copy of the ***marker_layer***, and may have attached properties depending on the mode specified in the options block.

- ***target_layer***

  A required argument that specifies the name of the layer with shapes to analyze. The feature vectors describe the information on this layer.

- ***marker_layer***

  A required argument that specifies the name of a layer with POI markers. For non-rectangular shapes, the POI is the center of the bounding box.

- **options *so_opt***

  A required keyword and argument indicating the name of the sonr_options block to use.

Description

The setlayer sonr_vector_capture command collects feature vectors and writes them to a database specified by out_db.

The following commands must appear in the sonr_options block:

- target to identify the layout.
- marker to identify which areas to capture.
- pv_mode to identify what data to measure.
- out_db to name the database of results.

If the modelpath or directory in which you started the run does not contain a directory called *lmod* that contains a file named *Lithomodel*, the recipe must also contain a litho_model specification.

Examples

This shows a complete setup file that would be called by SONR Vector Capture. The setlayer sonr_vector_capture command passes a layer called “output_layer” back to SONR Vector Capture.

```
modelpath ./
layer target_layer
layer marker_layer

sonr_options SONR_OPT {
   target target_layer
   marker marker_layer
   pv_mode 1
   litho_model my_mod
   max_sampling_distance 0.08
   out_db pv.db
}

setlayer output_layer = sonr_vector_capture target_layer marker_layer \
   OPTIONS SONR_OPT
```

The example setup file provides a litho model and explicit values for the sampling distance, but these are not required.

Related Topics

SONR Vector Capture

sonr_options

sonr_options

Type: Litho Setup File Commands

Command block within a setup file that defines the Calibre SONR settings.

Usage

sonr_options ***name*** **'{'**
  options_list
  '}'

Arguments

- ***name***

  A required argument naming the sonr_options block so that it may be referred to by other commands.

- ***options_list***

  A required list of settings for the Calibre SONR run surrounded by required brackets ({ }). The commands must appear each on their own line. The pound sign character (#) comments out the rest of the line.

Description

Creates a uniquely named sub-command block inside a litho setup file to set options for the sonr_analyze and sonr_vector_capture commands. The complete list of options is shown in "sonr_options Commands" on page 71.

Examples

This shows how a setlayer command refers to a sonr_options block:

```
sonr_options la.opt {
   target metal1
   marker vc_out

   mode cluster layout.db
   out_db clustered_layout.db
}

setlayer output_layer = sonr_analyze metal1 vc_out options la.opt
```

Related Topics

setlayer sonr_analyze

setlayer sonr_vector_capture

SONR_AUX_LAYER

Type: External setup file command

denseopc only. Defines auxiliary layers for SONR_COLLECT and SONR_PREDICT.

Usage

SONR_AUX_LAYER -tag *name* -mod *feature_list* -basemodel *optical_model*
[-densityLayer *layer*] [-dist *dbus*] [-layer *name*] [-prop *layer*] [-reduction *factor*]
[-outName *prefix*]

Arguments

- **-tag *name***

  A required argument identifying a tagged set of fragments that are considered when constructing feature vectors.

- **-mod *feature_list***

  A required argument identifying the list of features to extract, typically named *phv.mod*. Use the same file that was used for SONR_COLLECT or SONR_PREDICT…-mod.

  The setup file should also include an "ml_featurevector_load ***name file***" command, where the name is the same as ***feature_list***.

- **-basemodel *optical_model***

  A required argument specifying the optical model used for the nominal process window condition for this layer. This model may be the same as the optical model used by other layers, including the main layer in SONR_COLLECT or SONR_PREDICT, but must be identified.

  If the optical model is not in the litho model, use the Calibre nmOPC command optical_model_load to include it in the run.

- -densityLayer *layer*

  An optional argument that specifies an input layer from which to calculate density. By default, the layer specified by -layer is used.

- -dist *dbus*

  An optional argument that specifies how far from the fragment to search for the DFM property marker. Units are in dbu. The default distance is 60 dbu.

  This argument has no effect when -prop is not specified.

- -layer *name*

  An optional argument that identifies the layer containing the geometries from which to collect the feature vectors. If this argument is not specified, it defaults to a layer of type "opc." (See "layer (for denseopc_options Block)" in the *Calibre nmOPC User's and Reference Manual*.)

- -prop *layer*

  An optional argument that reads DFM properties from a layer. The properties are included as columns in the CSV and database files. All DFM properties within 60 dbu of the fragment on the layer are collected. (To set a different search range, use -dist.)

  The *layer* must match the ***layer*** declared in a read_layer_properties command. (See *Calibre nmOPC User's and Reference Manual*.)

- -reduction *factor*

  An optional argument specifying to simplify the model. Larger values reduce the computational load.

  0 — No reduction. This is the default.

  1 — Half reduction.

  2 — Quarter reduction.

  Only the values 0, 1, and 2 are supported. The -reduction in SONR_AUX_LAYER must match that used in SONR_COLLECT and SONR_PREDICT.

- -outName *prefix*

  An optional argument that specifies a prefix to use on column names for features derived from this layer. If this is not specified, the layer name is used.

Description

The SONR_AUX_LAYER command adds additional layers to the SONR_COLLECT or SONR_PREDICT operation. These commands can only be specified in a denseopc_options block in an external setup file.

Examples

This example adds the fragments in the hsFrags2 tag set on the layer target2 to an operation.

```
SONR_AUX_LAYER -layer target2 -tag hsFrags2 -mod nmod -basemodel optical2
```

Also see "Example 2" on page 66 in SONR_COLLECT for a longer example that shows supporting commands.

Related Topics

Collecting Multilayer Feature Vectors

SONR_COLLECT

Type: External setup file command

denseopc only. Collects features vectors and writes them to a database for creating a hotspot prediction model.

Usage

SONR_COLLECT -tag *name* -mod *model* [-addlayer *name* [-mttag *name*]]…
[-append [*database*]] [-basemodel *optical*] [-density *name*]… [-densityLayer *layer*]
[-dist *dbus*] [-layer *name*] [-prop *layer*] [-quantize *number*] [-record *values*]…
[-reduction *factor*] [-outdb *name*] [-outlayer *name*]

Arguments

- **-tag *name***

  A required argument identifying a tagged set of fragments that are considered when constructing feature vectors. These fragments should be on the layer with the mask geometry. If -layer is not specified, SONR_COLLECT uses the layer of type "opc."

  By default, only fragments that are part of the tag set contribute to the feature vectors. You can include tagged fragments on other layers in the feature vectors by using the -addlayer argument.

- **-mod *model***

  A required argument identifying the list of features to extract, typically named *phv.mod*. This serves as a template for the model. Ask your Siemens representative for a "seed model" reflecting the most recent best practices.

  The ***model*** identifier should match the name supplied in an "ml_featurevector_load ***name file***" command. (This command is documented in the *Calibre nmOPC User's and Reference Manual*.) Do not use the SONR Layout Analyze "model" command.

- -addlayer *name* [-mttag *name*]

  An optional argument to include tagged fragments from additional layers when collecting the feature vectors. These layers can be any layer type and should also appear in SONR_AUX_LAYER commands.

  -mttag *name* — Identifies a tag set on the layer. Fragments in the tag set along with fragments from **-tag *name*** are used in constructing the feature vectors. If -mttag is not specified, all fragments on the additional layer are used.

- -append [*database*]

  An optional argument that adds data from the current run to an existing database and writes the sum to -outdb. (Any existent data in the database specified by -outdb is overwritten.)

  The outcomes of the possible combinations are as follows:

  - -append A.db -outdb B.db

If *A.db* is a pre-existing database, the new data is appended to *A.db*. *B.db* is a superset of *A.db*. If *A.db* does not exist, *B.db* contains only the data from the current run.

Note

Be careful to not reuse a database name in -outdb when -append specifies a database; the original *B.db* is overwritten with *A.db* and the new run.

- -append -outdb C.db

  C.db is treated as the existing data. After the run, *C.db* contains both the original data and the new data.

- -basemodel *optical*

  An optional argument that identifies the optical model used for this process layer. If this argument is not specified, SONR_COLLECT uses the nominal optical model listed in the *Lithomodel* file.

- -density *name*

  An optional argument that identifies a PNG file from which to read precomputed density convolution values, typically for EUV flare. The file must also be named in a density_image_load command.

  The -density option can be specified more than once, though each name should be unique. Values are included in the feature vector.

- -densityLayer *layer*

  An optional argument that specifies an input layer from which to calculate density. By default, the layer specified by -layer is used.

- -dist *dbus*

  An optional argument that specifies how far from the fragment to search for the DFM property marker. Units are in dbu. The default distance is 60 dbu.

  This argument has no effect when -prop is not specified.

- -layer *name*

  An optional argument that identifies the layer containing the geometries from which to collect the feature vectors. If this argument is not specified, SONR_COLLECT uses the layer of type “opc.” (See “layer (for denseopc_options Block)” in the *Calibre nmOPC User’s and Reference Manual*.)

- -prop *layer*

  An optional argument that reads DFM properties from a layer. The properties are included as columns in the CSV and database files. All DFM properties on the layer are collected. The *layer* must match the ***layer*** declared in a read_layer_properties command. (See *Calibre nmOPC User’s and Reference Manual*.)

- -quantize *number*

  An optional argument that combines features with similar measurements to reduce the amount of memory used. The value indicates how many distinct feature values are retained; lower numbers reduce the memory used more than higher numbers.

  Use this argument when collecting feature vectors for training data on a large data set. A good initial value is 100.

- -record *values*

  An optional argument that adds one or more columns to the CSV and database files. The columns contain data from an annotated tag set. The -record argument can be specified multiple times.

  There are two different syntaxes for *values*:

  column_name tag — The keyword -record followed by the name for a column and the name of the tag set from which to populate it. Only a single column name-tag pair can be specified per -record keyword.

  (*column_name tag column_name tag* ...) — The keyword -record followed by parentheses enclosing a list of column name-tag pairs. There can be any number of pairs.

  These syntaxes can be combined. For example, any of these could be used:

  ```
  -record angle aTag -record angle_prev a1Tag -record angle_next a2Tag

  -record angle aTag -record (angle_prev a1Tag angle_next a2Tag)

  -record (angle aTag angle_prev a1Tag angle_next a2Tag)
  ```

  The columns contain property values from the specified tag set. If a fragment is not in *tag*, it receives a value of 0 in the column.

- -reduction *factor*

  An optional argument specifying to simplify the model. Larger values reduce the computational load.

  0 — No reduction. This is the default.

  1 — Half reduction.

  2 — Quarter reduction.

  Only the values 0, 1, and 2 are supported.

- -outdb *name*

  An optional argument that writes the collected feature vectors to a database named “*name*.” The default database name is *sonr_collect_out.db*.

  If the name specified in -outdb already exists, the file is overwritten.

- -outlayer *name*

  An optional argument that writes the results to a layer that can be returned via setlayer denseopc … MAP *name*.

Description

The SONR_COLLECT command captures the feature vector information for a layer. The results are saved to a database file for use in training the machine learning model for hotspot prediction. SONR_COLLECT also creates a seed model (*phvo.mod*) for use with sonr --cluster.

When collecting data for hotspot prediction models, use SONR Vector Capture to call an external setup file that includes SONR_COLLECT and setlayer denseopc instead of setlayer sonr_vector_capture. The external setup file includes Calibre nmOPC commands, but requires a Calibre SONR license and will not function when called from Litho DenseOPC.

Examples

Example 1

This example creates a database named *hp_single.db* from all the fragments on the opc layer, using the feature vectors listed in *phv.mod*.

```
## Most keywords in this file are documented in the
## Calibre nmOPC User’s and Reference Manual (calbr_nmopc_useref.pdf) or
## Calibre OPCverify User’s and Reference Manual (calbr_opcv_useref.pdf)
modelpath ./models
tilemicrons 15

layer target
layer marker

ml_featurevector_load nmod ./phv.mod

denseopc_options SONR_OPT {
    version 1
    layer target opc    mask_layer 0
    layer marker hidden mask_layer 0

    image lmod

    fragment_min 0.010
    fragment_max 0.030

#### In a real setup file, there is typically custom fragmentation
#### rules and tagging here.

    NEWTAG all target -out allFrags
    OUTPUT_SHAPE fragment allFrags allFrags 0.001 0.002

    SONR_COLLECT -tag allFrags -mod nmod -outdb hp_single.db
}

setlayer collect = denseopc target marker MAP target OPTIONS SONR_OPT
```

Example 2

This example shows how to include multiple layers, such as you might use for a multi-patterning layout. The layers target1, target2, and target3 represent a triple-patterning mask. They are all of type opc, as shown in the denseopc_options layer list:

```
layer target1 opc mask_layer 0 pattern 0
layer target2 opc mask_layer 0 pattern 1
layer target3 opc mask_layer 0 pattern 2
```

After fragmentation (not shown), the fragments are added to tag sets:

```
NEWTAG all target1 -out allFrags1
NEWTAG all target2 -out allFrags2
NEWTAG all target3 -out allFrags3
```

Typically more tag-based operations occur after this, such as creating tag sets of fragments within a hotspot marker or annotating a tag set with hotspot categories.

Before running SONR_COLLECT, the layers that will contribute to the feature vectors other than the main layer are identified with SONR_AUX_LAYER:

```
SONR_AUX_LAYER -layer target2 -tag hsFrags2 -mod nmod -basemodel optical2
SONR_AUX_LAYER -layer target3 -tag hsFrags3 -mod nmod -basemodel optical3
```

The SONR_COLLECT command then identifies the primary layer (-layer) and the additional fragments (-mttag and -addlayer) to create feature vectors, and that the output should be written to a database *multilayer.db* as well as the out_frags layer in the OASIS output.

```
SONR_COLLECT -layer target1 -tag hsFrags1 -mod nmod \
   -mttag hsFrags2 -mttag hsFrags3 -addlayer target2 -addlayer target3 \
   -outdb multilayer.db -outlayer out_frags
```

Related Topics

SONR_AUX_LAYER

Calibre nmOPC User’s and Reference Manual

SONR_PREDICT

Type: External setup file command

denseopc only. Analyzes a layout and predicts hotspots based on trained models.

Usage

SONR_PREDICT -tag *name* -mod *feature_list* -snr_mod *hotspot_model*
[-addlayer *name* [-mttag *name*]]… [-class *label*…] [-density *name*]… [-dist *dbus*]
[-gen *value*] [-layer *name*] [-prop *name*] [-prop_thresh *value*]
[-record *values*]… [-reduction *factor*][-outdb *name*]
[-dfm_property *type outlayer* {int | float} max]… [-outlayer *name*]

Arguments

- **-tag *name***

  A required argument identifying a tagged set of fragments that are considered when creating the calibration vectors. These fragments should be on the layer with the mask geometry. If -layer is not specified, SONR_PREDICT uses the layer of type "opc."

- **-mod *feature_list***

  A required argument identifying the list of features to extract, typically named *phv.mod*. Use the same file that was used for SONR_COLLECT …-mod.

  The setup file should also include an "ml_featurevector_load ***name file***" command, where the name is the same as ***feature_list***.

- **-snr_mod *hotspot_model***

  A required argument identifying the machine learning model trained with SONR_COLLECT vectors. Set ***hotspot_model*** to the name given in ml_model_load.

- -addlayer *name* [-mttag *name*]

  An optional argument to include tagged fragments from additional layers. These layers can be any layer type and should also appear in SONR_AUX_LAYER commands.

  -mttag *name* — Identifies a tag set on the layer. Fragments in the tag set along with fragments from **-tag *name*** are used in constructing calibration vectors. If -mttag is not specified, all fragments on the additional layer are used.

- -class *label*…

  An optional argument identifying the type of hotspot data that the tag set represents. The label should be one or more integers. To specify more than one label, separate the labels with a space like "-class 1 2 3".

  The -class option is ignored if not specified with -dfm_property.

- -density *name*

  An optional argument that identifies a PNG file from which to read precomputed density convolution values, typically for EUV flare. The file must also be named in a density_image_load command.

  The -density option can be specified more than once, though each name should be unique. Values are included in the feature vector.

- -dist *dbus*

  An optional argument that specifies how far from the fragment to search for the DFM property marker. Units are in dbu. The default distance is 60 dbu.

  This argument has no effect when -prop is not specified.

- -gen *value*

  An optional argument specifying a generation tag that will be attached to feature vectors added to an existing database. The -gen option is ignored if not specified with -dfm_property.

- -layer *name*

  An optional argument that identifies the layer containing the geometries from which to collect the calibration vectors. If this argument is not specified, SONR_PREDICT uses the layer of type "opc." (See "layer (for denseopc_options Block)" in the *Calibre nmOPC User's and Reference Manual*.)

- -prop *name*

  An optional argument that reads DFM properties from a layer. The properties are included in the database. All DFM properties on the layer are collected. The name must match the ***layer*** declared in a read_layer_properties command. (See *Calibre nmOPC User's and Reference Manual*.)

- -prop_thresh *value*

  An optional argument that specifies the threshold to use with -dfm_property Probability. The default value is 0.5.

- -record *values*

  An optional argument that adds one or more entries to the database. The entries contain data from an annotated tag set. The -record argument can be specified multiple times.

  There are two different syntaxes for *values*:

  entry_name tag — The keyword -record followed by the name for the entry and the name of the tag set from which to populate it. Only a single entry name-tag pair can be specified per -record keyword.

  (*entry_name tag entry_name tag* ...) — The keyword -record followed by parentheses enclosing a list of entry name-tag pairs. There can be any number of pairs.

  These syntaxes can be combined. For example, any of these could be used:

  ```
  -record angle aTag -record angle_prev a1Tag -record angle_next a2Tag
  ```

```
-record angle aTag -record (angle_prev a1Tag angle_next a2Tag)

-record (angle aTag angle_prev a1Tag angle_next a2Tag)
```

The entries contain property values from the specified tag set. If a fragment is not in *tag*, it receives a value of 0.

- -reduction *factor*

  An optional argument specifying whether to simplify the model. Larger values reduce the computational load.

  0 — No reduction.

  1 — Half reduction.

  2 — Quarter reduction.

  Only the values 0, 1, and 2 are supported.

- -outdb *name*

  An optional argument that writes the collected feature vectors to a database named "*name*." By default, no database is written.

- -dfm_property *type outlayer* {int | float} max

  An optional argument set that filters output based on the SONR_PREDICT results. The following property types are valid:

  Class — Use in conjunction with -class. Filtering is done based on the type of hotspot label. The output type should be set to int. Generally, there are two classes (hotspot and not hotspot). Requires a traditional SONR model.

  Gen — Use in conjunction with -gen. Filtering is done based on generation for incremental model building. Requires a traditional SONR model.

  Probability — Use in conjunction with -prop_thresh. Requires a fully supervised machine learning model (model_2) produced with sonr --model_creator.

  The -dfm_property argument can be specified multiple times per SONR_PREDICT statement. The *outlayer* value should match the layer name specified in -outlayer.

  Note

  The only arguments that can follow a -dfm_property argument set are another -dfm_property argument set or -outlayer.

- -outlayer *name*

  An optional argument that writes the results to a layer that can be returned via setlayer denseopc … MAP *name*.

Description

The SONR_PREDICT command predicts hotspots on the layout. The command automatically does vector capture and makes a prediction in one run.

SONR_PREDICT runs in external setup files that include Calibre nmOPC commands, but requires a Calibre SONR license. The external setup files must be referenced from a SONR Hotspot Predict operation and will not run when referenced from a Litho DenseOPC operation.

See "SONR Hotspot Predict" on page 31 for a complete example file.

sonr_options Commands

This section lists commands that can appear in a sonr_options block.

The OPTION block_name argument in the setlayer sonr_analyze and setlayer sonr_vector_capture commands specifies a sonr_options parameter block within the LITHO FILE block. Keywords in this block are case-insensitive.

The following keywords may appear in the sonr_options block, specified one per line:

Table 3-8. sonr_options Summary

Command	Description
append	Adds new feature vectors to an existing database.
cluster_method	sonr_analyze only. Formerly controlled how clustering is performed.
dist	sonr_vector_capture only. Specifies how far from the fragment to search for the DFM property marker.
label	sonr_vector_capture only. Tags feature vectors in the database with a numeric value that is informational only, and not used to determine clusters.
litho_model	sonr_vector_capture only. Identifies the location of the litho model for the process.
marker	sonr_vector_capture only. Required. Identifies the layer whose shapes indicate where to collect data.
max_sampling_distance	sonr_vector_capture only. Sets the maximum fragment length.
mode	sonr_analyze only. Required. Specifies whether to cluster or compare databases.
model	sonr_analyze only. Specifies the machine learning model to create or use.
normalize	sonr_analyze only. Normalizes the input data before training a model with it.
out_db	Required. Specifies a database name to receive the results.
poi	sonr_analyze only. Required. Identifies the layer containing points of interest.
prop	sonr_vector_capture only. Adds DFM property values to the feature vector in the database. The DFM properties are considered part of the feature vector value for clustering.
pv_mode	sonr_vector_capture only. Required. Specifies what properties to capture in the feature vectors.

Table 3-8. sonr_options Summary (cont.)

Command	Description
representative	sonr_analyze only. Flags one representative feature vector per cluster in the database.
target	Required. Identifies the layer containing layout geometry.

append

Type: sonr_options Commands

Adds new feature vectors to an existing database.

Usage

append [*existing_database*]

Arguments

- *existing_database*

  An optional name of a previously created database. The output of the current run will be written after the existing data. By default, the output is added at the end of the database specified in out_db.

  Caution

  If you specify *existing_database* and the out_db database is also an existing database, the out_db database is overwritten with the results of *existing_database* and the current run.

Description

Use append to collect the feature vectors of multiple runs into a single database. All runs must collect the same measurements (specified with pv_mode) and use the same normalization settings.

If Calibre SONR does not find *existing_database*, the run continues after writing the following message to the transcript:

```
DATABASE PROCESSING NOTE: database <filename> not found.
```

Verify spelling and that *existing_database* is in the directory from which the run was initiated. If the pv_mode, DFM Properties, or labels are different, the run exits with the following error:

```
PROCESS ML DATABASE: Database Specification not match (different fields
used), can't append database.
```

Examples

Example 1 — Appending and Keeping the Same Database Name

To write new feature vectors to an existing file (for example, *A.db*), add the command “append” to the setup file. Use *A.db* as the out_db setting.

```
out_db A.db
append
```

To append to the out_db, “append” does not require an argument.

This is the recommended usage because you do not risk losing any data in *A.db*.

Example 2 — Appending and Changing the Database Name

You can create a series of databases by providing the name of the Nth database to “append” and giving a new name to out_db:

```
out_db A_v5.db
append A_v4.db
```

In this case, the existing database is *A_v4.db*. The new database is *A_v5.db*. However, if the *A_v5.db* file already existed, the old contents are lost. At the next run, set append to *A_v5.db* and increment out_db to *A_v6.db*.

Related Topics

label

normalize

prop

pv_mode

cluster_method

Type: sonr_options Commands

sonr_analyze only. Formerly controlled how clustering is performed.

Note

As of version 2021.2, cluster_method is ignored. The command may remain in existing setup files for backwards compatibility.

Usage

cluster_method SONR1 ***setting***

Arguments

- ***setting***

  A required argument controlling the size of the clusters. Larger values result in more clusters. The recommended range is 8 to 128.

Description

Cluster_method is an optional layout analysis command. It is ignored by SONR Vector Capture. It creates clusters of feature vectors based on their similarity.

The SONR1 clustering method uses an unsupervised machine learning algorithm to cluster the data, based on the model generated using sonr --tree or sonr --cluster. The model is specified using the model command.

DFM properties that were added to the database with "prop" affect clustering results. Properties added with "label" do not.

Clusters are identified by the cluster ID property on shapes in the output layer.

Examples

```
mode cluster database_A
cluster_method sonr1 32
model model_A
```

Related Topics

sonr --cluster

sonr --tree

dist

Type: sonr_options Commands

sonr_vector_capture only. Specifies how far from the fragment to search for the DFM property marker.

Usage

dist *distance*

Arguments

- ***distance***

  A required argument that specifies how far from the fragment to search for a property. Units are in dbu. The default distance is 60 dbu.

Description

This optional command controls how far a property can be from a fragment to be detected. All markers of the appropriate type within this distance are included in the feature vector, so it is best to keep the distance small.

Related Topics

label

prop

label

Type: sonr_options Commands

sonr_vector_capture only. Tags feature vectors in the database with a numeric value that is informational only, and not used to determine clusters.

Usage

label ***property_layer property_name*** …

Arguments

- ***property_layer***

  A required argument specifying a layer name. The layer should have polygons with attached DFM properties.

- ***property_name***

  A required argument specifying one or more numeric DFM properties on the layer. DFM property names are case sensitive.

Description

There are two commands that read DFM properties and add them to the database with the feature vectors.

- Label adds the property as an identifier. Values do not affect similarity calculations.
- Prop adds the property as part of the feature vector. Values do affect similarity calculations.

The label data is attached at the end of each feature vector, but is not considered a feature.

When different property layers have identical property names, they are treated as the same property.

Examples

This example reads the property “hotspot” from the layer HS_NHS and labels the feature vectors with the value. The layer HS_NHS was created outside the example.

```
m1_opc    {COPY m1_opc}    DRC CHECK MAP m1_opc    OASIS 1
HS_NHS    {COPY HS_NHS}    DRC CHECK MAP HS_NHS    OASIS 2 PROPERTIES
target    {COPY target}    DRC CHECK MAP target    OASIS 101

target = SONR VECTOR CAPTURE m1_opc HS_NHS FILE svc.in MAP target_out

LITHO FILE svc.in [/*
  modelpath ./models

  layer m1_opc
  layer HS_NHS
```

```
      sonr_options SONR_OPT {
        target target_in
        marker marker0

        label HS_NHS hotspot

        pv_mode 1
        out_db results/vc_hotspot.db
      }

      setlayer marker0 = copy m1_opc

      setlayer target_out = sonr_vector_capture m1_opc marker0 HS_NHS \
                                        options SONR_OPT
    */]
```

Related Topics

prop

litho_model

Type: sonr_options Commands

sonr_vector_capture only. Identifies the location of the litho model for the process.

Usage

litho_model *directory*

Arguments

- ***directory***

  A required argument that specifies the directory containing a *Lithomodel* file and optical model(s). The directory can be a soft link.

Description

The litho_model command is required when feature vectors are to include non-geometric information such as intensity, which requires an optical model to calculate correctly.

The litho_model command requires that the outer setup file also include modelpath.

Examples

This shows a SONR Vector Capture setup, which loads a litho model from */site/process/models/lmod*.

```
frag_layer = SONR VECTOR CAPTURE target marker MAP sonr_out FILE vc.setup

LITHO FILE vc.setup [/*
   modelpath /site/process/models/
   layer target
   layer marker

   sonr_options vc.opt {
      target target
      marker marker
      pv_mode 2
      litho_model lmod
      ...
   }

   setlayer sonr_out = sonr_vector_capture target marker options vc.opt
*/]
```

Related Topics

modelpath [Calibre OPCverify User's and Reference Manual]

marker

Type: sonr_options Commands

sonr_vector_capture only. Required. Identifies the layer whose shapes indicate where to collect data.

Usage

marker *layer_name*

Arguments

- ***layer_name***

  A required argument that specifies a layer name.

Description

Any shapes on the target layer that interact with the shapes on the marker layer have their edges split into fragments. Each of these fragments that overlaps the marker layer shapes receives a POI that serves as a seed for the feature vector. These fragments are written to the output of the setlayer sonr_vector_capture command, which is then used by setlayer sonr_analyze as the poi layer.

The output layer is a copy of the marker layer, sometimes with results attached as properties depending on the mode.

Examples

```
marker metal1
```

max_sampling_distance

Type: sonr_options Commands

sonr_vector_capture only. Sets the maximum fragment length.

Usage

max_sampling_distance ***microns***

Arguments

- ***microns***

  A required argument specifying the maximum length to use when breaking edges into fragments.

Description

SONR Vector Capture breaks edges greater than max_sampling_distance into fragments. The fragments that interact with the marker layer receive a POI at their center. This command sets the maximum length of these fragments.

If you specify max_sampling_distance, be sure that the value is an integer multiple of the layout's database unit.

The valid ranges are as follows:

- **DUV**: 0.060 <= max_sampling_distance <= 0.200. (Default: 0.200)
- **EUV**: 0.020 <= max_sampling_distance <= 0.100. (Default: 0.100)

Examples

This sets the maximum fragment length to 0.08 microns, or 80 nanometers:

```
max_sampling_distance 0.08
```

mode

Type: sonr_options Commands

sonr_analyze only. Required. Specifies whether to cluster or compare databases.

Usage

Syntax 1

mode cluster *database*

Syntax 2

mode clusterThenDiff *database1 database2* BnotA

Arguments

- **cluster**

  A keyword that causes the run to analyze the feature vectors in database for similarity. The feature vectors, which each represent a particular POI's neighborhood, are grouped into similar sets and each set is assigned a cluster ID.

- **clusterThenDiff**

  A keyword that causes the run to cluster two different databases and then compare the clusters for similarity. Clusters are "binned" as being in just ***database1***, just ***database2***, or in both databases.

- ***database***

  database1 database2

  A required database or pair of databases, depending on the mode specified. The databases contain feature vectors generated by SONR Vector Capture. The databases must have been generated using the same pv_mode settings or the run exits with an error.

- **BnotA**

  A required keyword for Syntax 2. It specifies that the results contain those clusters that are only in ***database2***.

Description

The mode command is required for SONR Layout Analyze. It controls the run operation.

The cluster mode is similar to the command-line application, sonr --cluster. Both produce clustered databases that can be used for comparing layouts.

Examples

Example 1- Finding New Patterns

To identify patterns in a layout that were not previously seen, use clusterThenDiff and output the unique patterns from the new layout. In the following example, "pvA.db" represents the database of feature vectors from the first layout.

```
layer target
layer vc_out

sonr_options diff.opt {
   target target
   poi    vc_out

   mode clusterThenDiff pvA.db pvB.db BnotA
   out_db clustered_BnotA.db
}

setlayer sonr_out =  sonr_analyze target vc_out OPTIONS diff.opt
```

Related Topics

model

model

Type: sonr_options Commands

sonr_analyze only. Specifies the machine learning model to create or use.

Usage

model ***filename***

Arguments

- ***filename***

  A required argument identifying the machine learning model to use for layout analysis.

Description

The model specification identifies the machine learning model to use for this layout analysis. The model is generated using the sonr utility and feature vectors captured by SONR Vector Capture. The model is used for clustering with "mode cluster" or "mode clusterThenDiff."

Examples

The model specification is highlighted in the options block for a training run:

```
sonr_options train.opt {
   target target
   poi set_A

   model model_A
}
```

After the run completes, the directory contains a file, *model_A*, that can then be used when clustering layouts that will use the same manufacturing process.

Related Topics

Model Generation Commands

normalize

Type: sonr_options Commands

sonr_analyze only. Normalizes the input data before training a model with it.

Usage

normalize

Arguments

None.

Description

This optional keyword causes sonr_analyze to normalize the input data in the feature vector database when training a model. Later runs using that model must also include “normalize” in the sonr_options block.

Examples

This sonr_options block indicates to normalize the data in *pvA.db* when training model_A.

```
layer target
layer vc_out

sonr_options train.opt {
   target target
   poi    vc_out

   model model_A
   normalize
}

setlayer sonr_out = sonr_analyze target vc_out OPTIONS train.opt
```

Related Topics

mode

out_db

Type: sonr_options Commands

Required. Specifies a database name to receive the results.

Usage

out_db *filename*

Arguments

- ***filename***

  A required argument specifying a name for the database. The database will be created in the current working directory unless an absolute path is given.

  It is recommended to establish a naming convention for your databases, but this is not enforced. The Calibre SONR manual uses “*.db*” on all database names.

Description

This required command specifies where to save the results of SONR Vector Capture or SONR Layout Analyze. If you specify the name of a file that already exists, it is overwritten.

Examples

This example shows using an absolute path:

```
out_db /server100/project/full_chip.db
```

poi

Type: sonr_options Commands

sonr_analyze only. Required. Identifies the layer containing points of interest.

Usage

poi ***layer_name***

Arguments

- ***layer_name***

  A required argument that specifies a layer name.

Description

The poi keyword identifies a layer produced by SONR Vector Capture with small markers that indicate where feature vectors were collected. The output layer of sonr_analyze is a copy of the poi layer with properties attached.

Note

In version 2020.1, setlayer sonr_analyze used the marker keyword instead.

Examples

```
poi vc_out
```

prop

Type: sonr_options Commands

sonr_vector_capture only. Adds DFM property values to the feature vector in the database. The DFM properties are considered part of the feature vector value for clustering.

Usage

prop ***property_layer property_name*** …

Arguments

- ***property_layer***

  A required argument specifying a layer name. The layer should have polygons with attached DFM properties.

- ***property_name***

  A required argument specifying one or more numeric DFM properties on the layer. DFM property names are case-sensitive.

Description

There are two commands that read DFM properties and add them to the database with the feature vectors.

- Label adds the property as an identifier. Values do not affect similarity calculations.
- Prop adds the property as part of the feature vector. Values do affect similarity calculations.

Each property layer should appear in only one prop command. Each run can accept up to 20 property layers and 100 DFM properties.

When different property layers have identical property names, they are treated as the same property.

Examples

This example reads the properties bridge_min and bridge_max from the layer bridges and includes the values of the properties in the feature vectors. The layer bridges was created outside the example using Calibre OPCverify and DFM Property.

```
m1_opc    {COPY m1_opc}     DRC CHECK MAP m1_opc    OASIS 1
bridges   {COPY bridges}    DRC CHECK MAP bridges   OASIS 2 PROPERTIES
target    {COPY target}     DRC CHECK MAP target    OASIS 101

target = SONR VECTOR CAPTURE m1_opc bridges FILE svc.in MAP target_out

LITHO FILE svc.in [/*
  modelpath ./models
```

```
    layer m1_opc
    layer bridges

    sonr_options SONR_OPT {
      target target_in
      marker marker0

      prop bridges bridge_min bridge_max

      pv_mode 1
      out_db results/vc_bridge_prop.db
    }

    setlayer marker0 = copy m1_opc

    setlayer target_out = sonr_vector_capture m1_opc marker0 bridges \
                                            options SONR_OPT
  */]
```

Related Topics

label

pv_mode

Type: sonr_options Commands

sonr_vector_capture only. Required. Specifies what properties to capture in the feature vectors.

Usage

pv_mode {**1** | **2**}

Arguments

- **1**

  Only geometric features such as layout dimensions and density are measured.

- **2**

  Measures geometric features and image qualities, such as image intensity based on nominal and process window conditions, and mask and wafer geometry.

Description

This command controls what properties are measured and recorded in the feature vectors.

If pv_mode 2 is specified, the input layout simulates image results. In this case, it is recommended to also specify litho_model so that the fracturing is based on the correct optical model. If litho_model is not specified, the program looks for an *lmod* directory containing a *Lithomodel* file in the modelpath and uses the first one.

If the sonr_options block includes prop or label, associated DFM properties are included in the feature vectors.

Examples

```
pv_mode 1
```

Related Topics

label

prop

representative

Type: sonr_options Commands

sonr_analyze only. Flags one representative feature vector per cluster in the database.

Usage

representative

Arguments

None.

Description

When “representative” is used with “mode cluster” or “mode clusterThenDiff”, the SONR Layout Analyze run adds a property named “representative” to all feature vectors in the output database. One feature vector per cluster has the “representative” property set to 1, and the others have it set to 0.

This optional command has no effect in SONR Vector Capture runs.

Examples

The following lines are part of a SONR Layout Analyze setup file. The run is instructed to cluster the vectors in databases *pvA.db* and *pvB.db*, then compare. The clusters that are in B but not A are written to the result database *clustered_BnotA.db*. Each cluster has one vector marked as “representative.”

```
mode clusterThenDiff pvA.db pvB.db BnotA
model model.A
out_db clustered_BnotA.db
representative
```

Related Topics

mode

target

Type: sonr_options Commands

Required. Identifies the layer containing layout geometry.

Usage

target *layer_name*

Arguments

- ***layer_name***

  A required argument that specifies a layer name. This layer must be a polygon layer.

Description

The target specification is always required in a sonr_options block. It identifies layers containing the geometry to work upon. It can be specified more than once per options block.

Examples

```
target via12
```

Index

Third-Party Information

Details on open source and third-party software that may be included with this product are available in the *<your_software_installation_location>/legal* directory.