SIEMENS EDA

Calibre® MDPDefectAvoidance™ User's Manual

Software Version 2021.2

SIEMENS

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

Table of Contents

List of Figures

List of Tables

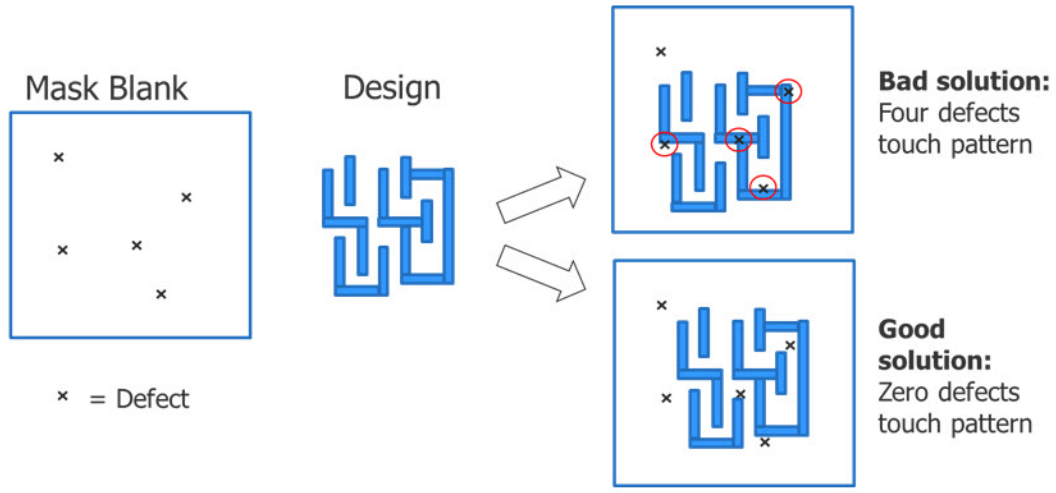
Chapter 1
Introduction to Calibre MDPDefectAvoidance

Calibre® MDPDefectAvoidance™, a part of the Calibre Mask Data Preparation (MDP) suite of tools, minimizes the impact of defective substrates in mask manufacturing by avoiding the defects from being placed on pattern shapes of the design.

The Calibre MDPDefectAvoidance solution rotates the blank (with its defect data) and shifts the blank (defect data) in X and Y directions such that no critical defect lies on a pattern on the mask. After the shift and rotation of the blank, if defects overlapping with mask patterns still remain, non-critical defects are removed in increasing order of criticality and non-critical mask pattern layers are ignored to find acceptable solutions.

Mask blank shifting is required to find solutions around potential defect impact (as shown in Figure 1-1).

Figure 1-1. Minimizing Impact of Defects



The Blank-Design Pairing Optimizer of Calibre MDPDefectAvoidance maximizes the number of manufacturable masks given the blanks and designs. Once the best solutions for individual blank-design pairings have been determined using Calibre MDPDefectAvoidance, the pairing optimizer finds the best blank-design pairs.

Defect Avoidance Overview

The EUV substrate is a stack comprised of approximately 40 alternating silicon and molybdenum silicide layers covered with an absorber layer and a capping material. Complex mask topology (as compared to chrome on glass (COG) or attenuated mask substrates) leads to an extended deposition process during blank fabrication. This increases the probability of introducing a greater number of defects.

Direct repair of the multilayer stack is not possible; removing distortions or extraneous material requires a highly-accurate local deposition of a multilayer reflection stack. The impact of a mask defect on wafer printing results could manifest itself as pinching and bridging, missing features, and local CD variations.

Due to these issues, the layout must be designed so defects do not impact printed results. Using Calibre MDPDefectAvoidance, you find the tolerance by which layout data should be shifted to prevent defects from impacting features. If a solution is not available by considering all defects, then you can filter or ignore non-critical defects based on their size, classification, or the layer on which they appear (currently supported only for MEBES job decks and OASIS®[1] formats) to find the solution space.

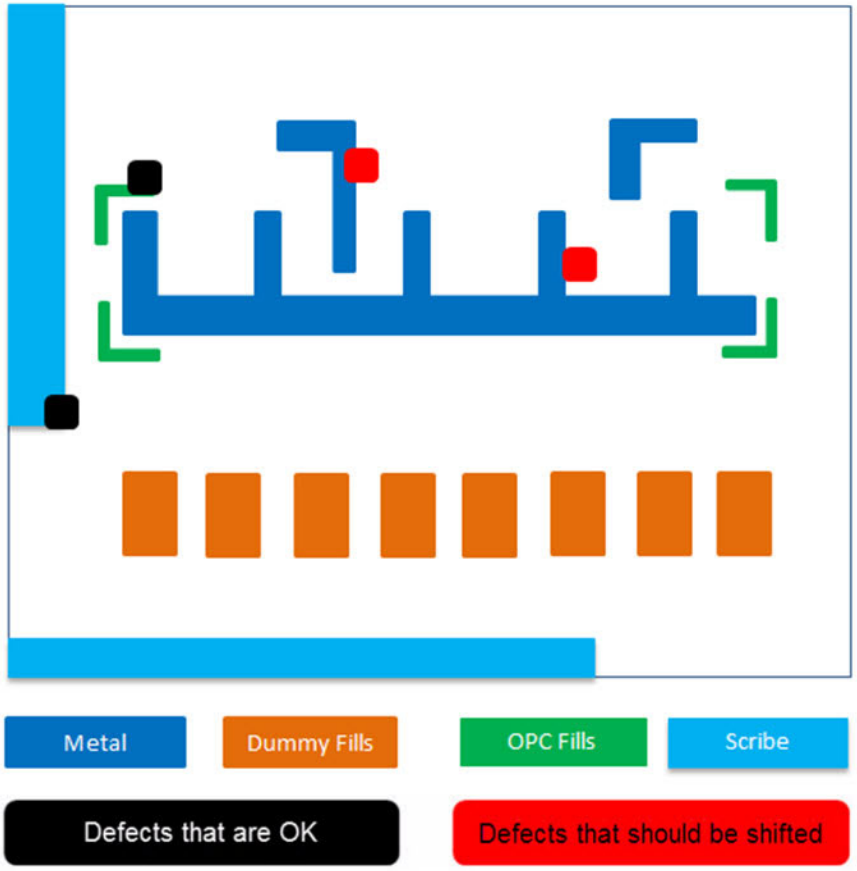
Layer Priorities

For MEBES job decks and OASIS design formats, you can prioritize the defects based on the layer geometries they appear on. For example, defects on dummy fills or scribes may not be as critical as defects landing on metals. The layer priorities functionality assumes that the input layouts are split out into different layers or levels based upon their functional intent (for example, metals could be in level 1, dummy fills on level 10, OPC fills on level 15, and so on in a MEBES job deck). The tool does not perform processing to separate out the layers.

As illustrated in Figure 1-2, the defects that appear on OPC fills and scribes have no impact, but the defects that appear on the metal must be shifted.

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

Figure 1-2. Multiple Layers in Design



You can specify priority of layers and defect size tolerance for that layer in parameter file or from the Calibre MDPDefectAvoidance GUI.

For example, you could specify that if a defect appears on a metal layer and the size of the defect is greater than 0.025 microns (25 nm), then that defect is avoided.

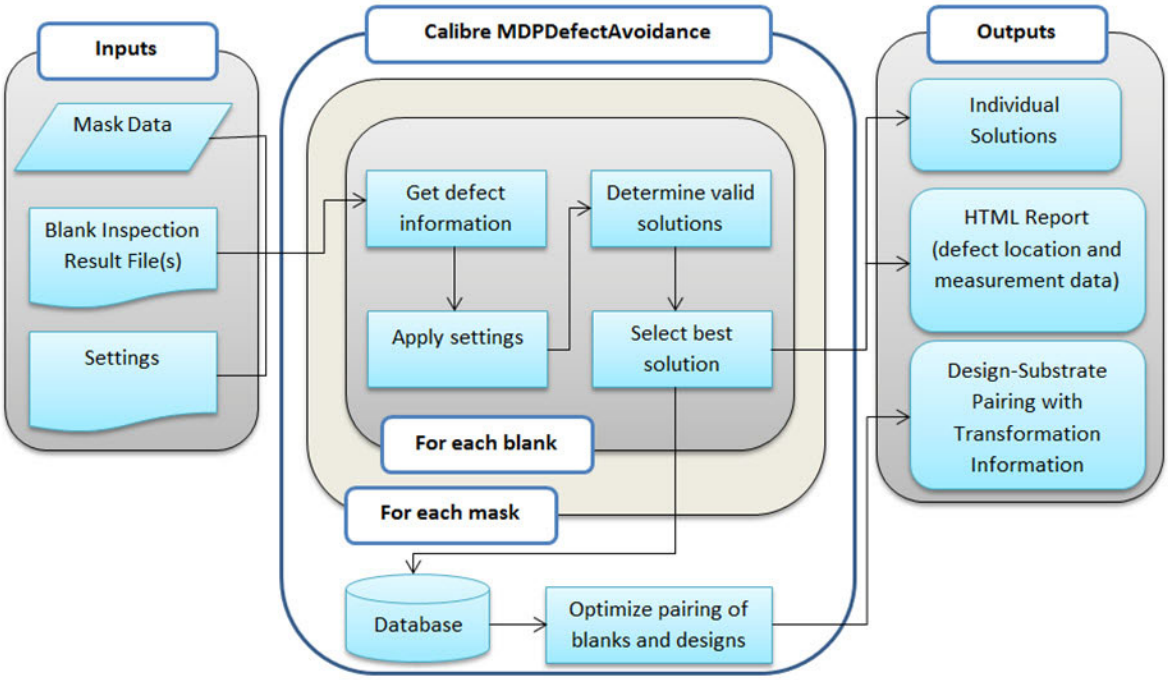
Refer to the layer priorities parameters description in “Parameters File Format” on page 87 for further information.

Calibre MDPDefectAvoidance Workflow

The Calibre MDPDefectAvoidance tool is launched through the Calibre® MDPview™ tool.

Figure 1-3 illustrates the Calibre MDPDefectAvoidance flow.

Figure 1-3. Calibre MDPDefectAvoidance Flow



The Calibre MDPDefectAvoidance is summarized with these steps:

1. Invoke Calibre MDPDefectAvoidance through Calibre MDPview (see "Calibre MDPDefectAvoidance Modes of Operation" on page 17) including a parameters text file containing control settings (see "Parameters File Format" on page 87 for further information on the parameters file).
2. In the Calibre MDPDefectAvoidance GUI, specify inputs for the defect avoidance run. This includes specifying the mask data and blank inspection results files (see "Calibre MDPDefectAvoidance Inputs" on page 60 for further information).
3. Run Calibre MDPDefectAvoidance to generate the following output:
   - Results of a single run in a text file
   - Blank-Design database file
   - Visualization Relational Database (RDB) file
   - HTML report

   These results are described in further detail in "Calibre MDPDefectAvoidance Outputs" on page 62.
4. Load the visualization RDB database file into Calibre RVE to view the output results.
5. Use the Blank-Design database file to perform blank-design pairing operations.

This overall flow is illustrated further in "Using Calibre MDPDefectAvoidance" on page 23.

Calibre MDPDefectAvoidance is part of a suite of tools for the Calibre Mask Data Preparation (MDP) flow. Calibre MDP provides seamless data manipulations required for mask data format conversion in one batch run, extending hierarchical representations of data.

Table 1-1 lists the documents associated with their related Calibre tools.

Table 1-1. Related Products and Their Manuals

Related Products	Documentation
Calibre® FRACTUREc™ Calibre® FRACTUREh™ Calibre® FRACTUREi™ Calibre® FRACTUREj™ Calibre® FRACTUREm™ Calibre® FRACTUREn™ Calibre® FRACTUREp™ Calibre® FRACTUREt™ Calibre® FRACTUREv™ Calibre® MDPmerge™ Calibre® MDPstat™ Calibre® MDPverify™ Calibre® MPCpro™ Calibre® MASKOPT™ Calibre® MDP Embedded SVRF	*Calibre Mask Data Preparation User's and Reference Manual* *Calibre Release Notes*
Calibre® MDPview™	*Calibre MDPview User's and Reference Manual* *Calibre Release Notes*
Calibre® Interactive™ Calibre® RVE™	*Calibre Interactive User's Manual* *Calibre RVE User's Manual*
Calibre® nmDRC™ Calibre® nmDRC-H™	*Calibre Release Notes* *Calibre Verification User's Manual* *Standard Verification Rule Format (SVRF) Manual*
Calibre® WORKbench™	*Calibre WORKbench User's and Reference Manual*
Tcl/Tk Batch Commands	*Calibre DESIGNrev Reference Manual*

Table 1-1. Related Products and Their Manuals (cont.)

Related Products	Documentation
Calibre® Metrology API (MAPI)	*Calibre Metrology API (MAPI) User's and Reference Manual*
Calibre® Job Deck Editor	*Calibre Job Deck Editor User's Manual*
Calibre® MDPDefectAvoidance™	*Calibre MDPDefectAvoidance User's Manual*
Calibre® nmMPC™ Calibre® nmCLMPC	*Calibre nmMPC and Calibre nmCLMPC User's and Reference Manual*
Calibre® MPCverify	*Calibre MPCverify User's and Reference Manual*
Calibre® DefectReview™	*Calibre DefectReview User's Manual*
Calibre® MDPAutoClassify™	*Calibre MDPAutoClassify User's Manual*
Calibre®DefectClassify™	*Calibre DefectClassify User's Manual*

Calibre MDPDefectAvoidance Prerequisites

There are prerequisites for Calibre MDPDefectAvoidance.

- **Platform support** — Calibre MDPDefectAvoidance is supported on the AOI platform. Refer to the *Calibre Administrator's Guide* for instructions on how to install Calibre software.
- **Licensing** — To run Calibre MDPDefectAvoidance, you must have the license file required by Calibre MDPview. To visualize the defect locations after shifts in Calibre MDPView or to create an HTML report, you must have a Calibre RVE license. For more information on licensing, refer to the *Calibre Administrator's Guide*.
- **Environment Variables** — Before invoking the application, you must set the environment variable MGC_HOME to the location of your Siemens EDA tree. Calibre tools require that the MGC_HOME environment variable be set. See the *Calibre Administrator's Guide* for details.

Calibre MDPDefectAvoidance Modes of Operation

You run the Calibre MDPDefectAvoidance from a command line shell using command line invocations, or invoke it from Calibre MDPview.

Command Line Mode

You can run Calibre MDPDefectAvoidance from a command console. The actual command you use controls which of the tools you invoke. The command arguments control the mode of operation.

Procedure

1. Invoke a command line shell.
2. In the shell, invoke the application by using the following syntax:

   There are three different syntax forms. The first is for initiating a Calibre MDPDefectAvoidance run:

   ```
   calibremdpv [-remotefile filename] -a da avoid parameter_file
     [remotemachinesfile remote_machines_filename [-skipda] ]
   ```

   where:

   - -remote_file *filename* — An optional parameter that loads a file containing information about remote machines to execute Calibre MDPDefectAvoidance in distributed mode using Calibre® MTFlex™ architecture. For more information on Calibre MTFlex, refer to the *Calibre Administrator's Guide*.
   - da avoid *parameter_file* — A required argument that initiates a defect avoidance run. The *parameter_file* is a text file that contains the control settings. See "Calibre MDPDefectAvoidance Files" on page 86.

     You can view the complete Calibre MDPDefectAvoidance command line options by using the following command in a shell:

     ```
     calibremdpv -a da help
     ```

The usage line displayed:

```
Usage:
Defect Avoidance: da avoid <parameter_file>
[-configfile <configuration_file>] [-remotemachinesfile
<remotemachinesfilename> [-skipda] ]

Pairing Optimizer: da optimize <database_file> [-paramfile
optimizer_parameter_file] [-outputfile output_file]

DA Verify: da verify <darun_parameter_file>
<darun_output_result_dir> [-configfile
<darun_configuration_file>]

Parallel Html Report: da htmlreport <darun_parameter_file>
<darun_output_result_dir> -remotemachinesfile
<remotemachinesfile> [-configfile
<darun_configuration_file>]
```

- remotemachinesfile *remote_machines_filename* — An optional parameter that loads a file containing information on remote machines. The remote machines specified in the file are used for parallel indexing of OASIS and MEBES chips available inside a job deck when JOBDECK is specified as Mask Data Type input. This is also used for parallel execution of blank inspection reports when JOBDECK is specified as the Mask Data Type. This parameter is also used for parallel execution of HTML report generation. For details, refer to the Remote Information description in “Specifying Defect Avoidance Run Settings” on page 31.

- -skipda — An optional parameter used along with -remotemachinefile parameter. If -skipda is specified, the Calibre MDPDefectAvoidance run is stopped after index file generation and no functions of the Calibre MDPDefectAvoidance run are performed. This keyword is supported only in the batch mode of Calibre MDPDefectAvoidance.

  Siemens EDA recommends that index and bin files be generated prior to defect avoidance runs. For job decks with multiple chips, this can be achieved using the following command:

  ```
  calibremdpv -a da avoid parameter_file  \
     [-remotemachinesfile remotemachinesfilename [-skipda] ]
  ```

  This example consumes one Calibre MDPDefectAvoidance license, but Calibre MDPview license consumption depends on the number of remote machines and cores available on remote machines.

  The -skipda option in conjunction with the -remotemachinefile keyword is used for index and bin file generation when using job decks. Its purpose is to consume only one Calibre MDPDefectAvoidence license for index and bin file generation prior to a Calibre MDPDefectAvoidance run. If -skipda is used without -remotemachine file, an invalid command error message is issued.

The following applies:

- HTML Reports are also generated in parallel during a defect avoidance run if the create_html_report parameter is set to true in the Calibre MDPDefectAvoidance parameters file.
- Index files must be generated prior to the Calibre MDPDefectAvoidance run.
- If an index file for a chip exists, it is not regenerated. If chip files are modified, related index files must be deleted to allow index regeneration to occur.
- When OASIS index files are generated, viewer components are not available.

Invoking from Calibre MDPview

You can invoke Calibre MDPDefectAvoidance from Calibre MDPview.

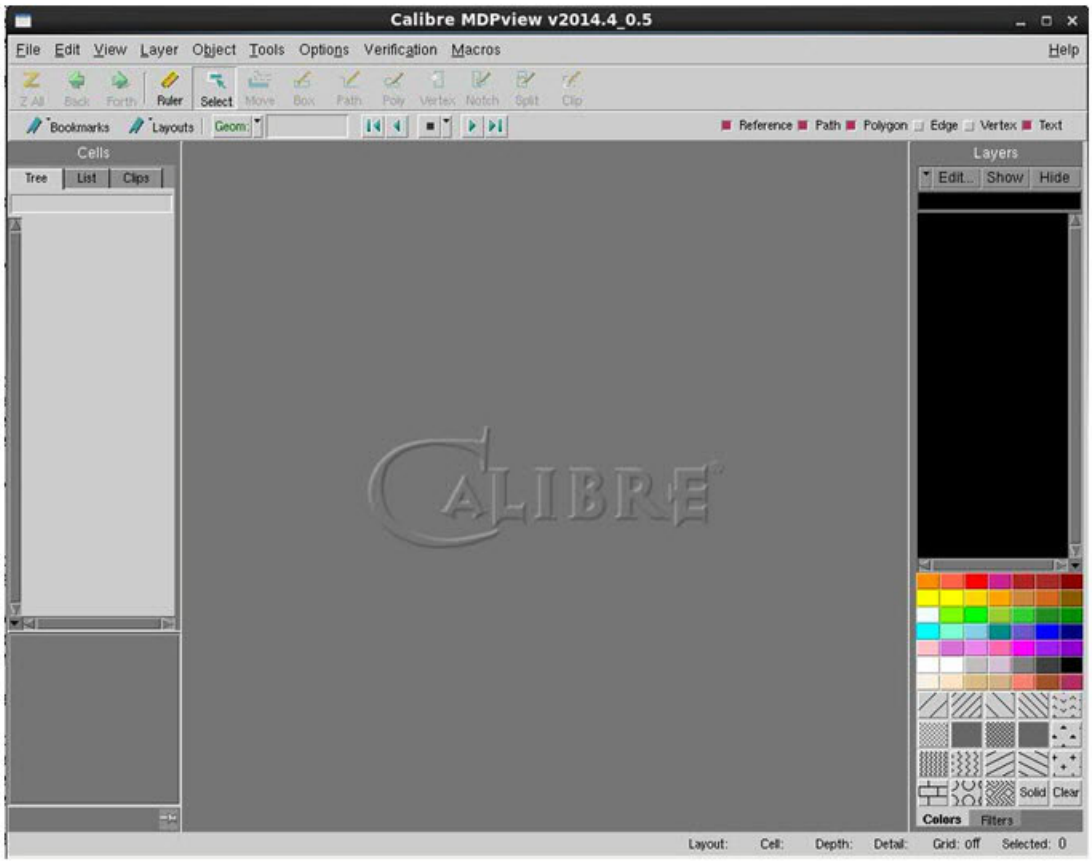
Procedure

1. Invoke Calibre MDPview a command shell using the following syntax.

```
calibremdpv [-remotefile remote_file]
```

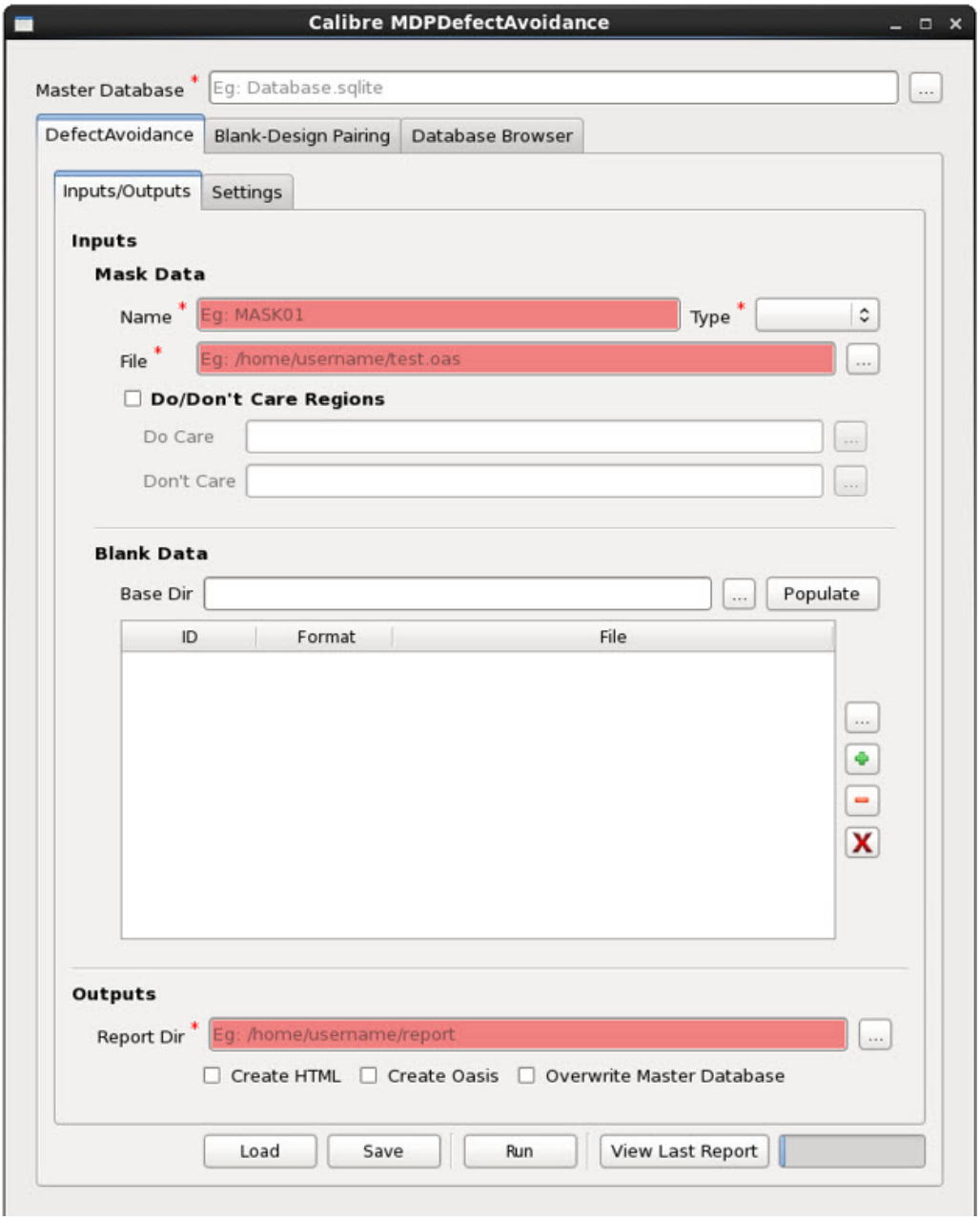
Calibre MDPview appears.

Figure 1-4. Calibre MDPview



2. In Calibre MDPview, select **Tools > MDPDefectAvoidance**. The Calibre MDPDefectAvoidance dialog box appears.

Figure 1-5. Calibre MDPDefectAvoidance Dialog Box



3. Once the GUI is launched, specify required inputs.
4. Click **Run** to start the defect avoidance run.
5. Refer to "Using Calibre MDPDefectAvoidance" on page 23 for an overview of its usage.

Syntax Conventions

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

Table 1-2. 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]	

Chapter 2
Using Calibre MDPDefectAvoidance

Calibre MDPDefectAvoidance is launched through Calibre MDPview and all major defect avoidance operations are performed through the Calibre MDPDefectAvoidance interface.

Calibre MDPDefectAvoidance Use Flow

The Calibre MDPDefectAvoidance use flow can be summarized in a series of steps.

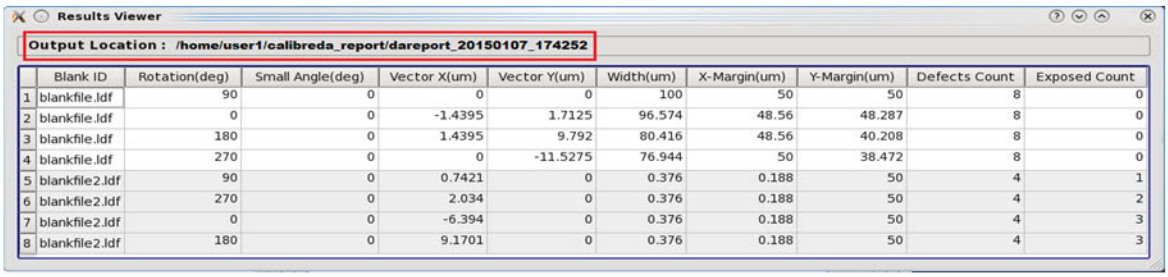
Procedure

1. Invoke Calibre MDPDefectAvoidance from Calibre MDPview (see "Calibre MDPDefectAvoidance Modes of Operation" on page 17).
2. Specify inputs such as the mask data, blank mask inspection data, and do and do-not-care region files. See "Specifying Inputs and Outputs" on page 25.
3. Specify defect avoidance run outputs such as the results file, visualization database file, and report directory. See "Specifying Inputs and Outputs" on page 25.
4. Specify defect avoidance run settings such as run settings, including defect placement, defect priorities, mask data layer rules (for layer priorities), and filtering out defects by size, area, or type. See "Specifying Defect Avoidance Run Settings" on page 31.
5. Save parameters for loading in future sessions. See "Saving and Loading a Parameter File" on page 40.

6. Once you have set all parameters, click **Run**. Once the run completes, results appear in a Results Viewer window.

   The results file, visualization database file, and HTML report are saved to the Output Location, as indicated in Figure 2-1.

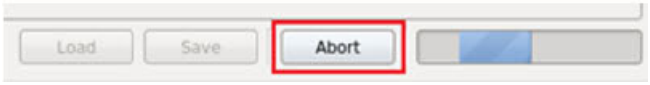
Figure 2-1. Results Viewer Output Location



	Blank ID	Rotation(deg)	Small Angle(deg)	Vector X(um)	Vector Y(um)	Width(um)	X-Margin(um)	Y-Margin(um)	Defects Count	Exposed Count
1	blankfile.ldf	90	0	0	0	100	50	50	8	0
2	blankfile.ldf	0	0	-1.4395	1.7125	96.574	48.56	48.287	8	0
3	blankfile.ldf	180	0	1.4395	9.792	80.416	48.56	40.208	8	0
4	blankfile.ldf	270	0	0	-11.5275	76.944	50	38.472	8	0
5	blankfile2.ldf	90	0	0.7421	0	0.376	0.188	50	4	1
6	blankfile2.ldf	270	0	2.034	0	0.376	0.188	50	4	2
7	blankfile2.ldf	0	0	-6.394	0	0.376	0.188	50	4	3
8	blankfile2.ldf	180	0	9.1701	0	0.376	0.188	50	4	3

7. To safely stop a defect avoidance run, click the **Abort** button, as shown in the following figure:

Figure 2-2. Abort Button



   A dialog box appears that asks for confirmation of the abort. If you click **Yes**, the run is aborted. If you click **No**, the run continues. If no action is taken before the run completes, then the dialog box closes and either the run results or an error window is displayed.

   Note the following:

   - To ensure that a defect avoidance run is aborted safely, it may take some time to stop a run. For example, if an abort is attempted during design loading, the run stops once loading is complete. If it takes a longer period of time, the following message is displayed:

     ```
     Aborting...
     This may take some time. Please wait.
     ```

   - All GUI widgets except the **Abort** button are deactivated when the defect avoidance execution is in progress. The **Abort** button is deactivated once you confirm the abort. Widgets are enabled once the run is completed or aborted.

   - If a run is aborted, the **View Last Report** button is not displayed.

8. Use Calibre RVE to view the defects on the layout. See “Using Calibre RVE for Output Visualization” on page 41.

9. Once the best solutions for individual blank-design pairings have been determined using defect avoidance, use the Blank-Design Pairing Optimizer to optimize across all blanks and designs (see “Using the Blank-Design Pairing Optimizer” on page 44).

The following requirements are assumed for this flow:

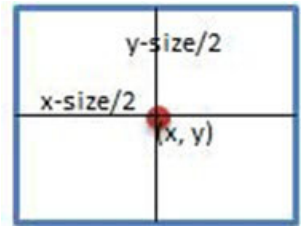
- The mask/layout data is aligned with the blank inspection data as follows:
  - The center of the mask/layout data is aligned with the center of the blank inspection data where:
    - The center of the mask/layout data corresponds to the center of the bounding box of the mask/layout data.
    - The center of the blank extracted as follows:

      For LDF and text formats, the origin of the defect file (0, 0) is considered as the center of the defect file and aligned with the center of the input data.

      For KLARF, the SampleCenterLocation value is considered the center and aligned with the center of the input data. Defect coordinates are calculated based on SampleCenterLocation and the first AlignmentPoints value (available in inspection file) is calculated as follows:

      X_Relative = X_Value (from the inspection file) + AlignmentPoint_X – SampleCenterLocation_X
- The defect coordinates specified in the blank file are considered to be the center of the defect as shown in Figure 2-3. The X and Y size are read from the blank inspection report files.

Figure 2-3. Defect Location



10. Use the DAverify utility to automatically cross check the correctness of solution provided by a defect avoidance run (see “Calibre MDPDefectAvoidance Use Flow” on page 23).

Specifying Inputs and Outputs

You specify inputs and outputs from the **Inputs/Outputs** tab of the Calibre MDPDefectAvoidance dialog box.

For more information on inputs and outputs, refer to “Calibre MDPDefectAvoidance Inputs” on page 60 and “Calibre MDPDefectAvoidance Outputs” on page 62.

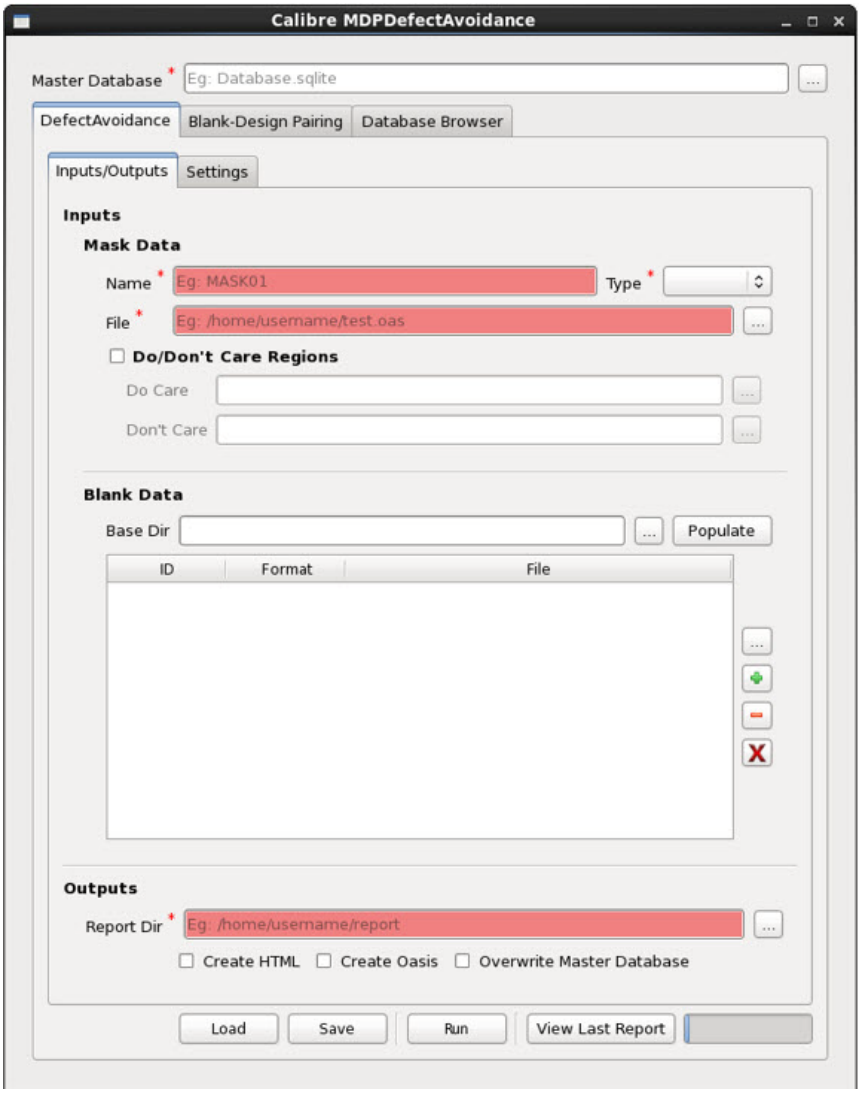
Procedure

1. Invoke Calibre MDPDefectAvoidance from Calibre MDPview (see "Calibre MDPDefectAvoidance Modes of Operation" on page 17).
2. At the top of the Calibre MDPDefectAvoidance window, specify the location where the databases are to be generated in the **Master Database** entry field (or click the Browse (**...**) button to locate the path).

Figure 2-4. Master Database

3. In the Calibre MDPDefectAvoidance dialog box, click the **Inputs/Outputs** tab.

Figure 2-5. Calibre MDPDefectAvoidance Dialog Box (Inputs/Outputs Tab)



4. In the **Inputs/Outputs** tab, specify the input types:

   - Mask Data

Figure 2-6. Input Mask Data

   i. Select a mask type from the Type pull-down menu, and then click the **Browse (...)** button to locate a mask file.

   ii. Specify a unique mask identifier in the Name entry field.

      - You can specify Mask Data Layers if the Type is OASIS® or MEBESJOB. For a MEBES job deck, specify layer numbers. For OASIS, specify the data type along with a layer number. Refer to the mask_data_layer parameter in "Parameters File Format" on page 87 for further details.

Figure 2-7. Mask Data Layers

      - If the Type is VSBJOB, you must specify a Layout Name as well.

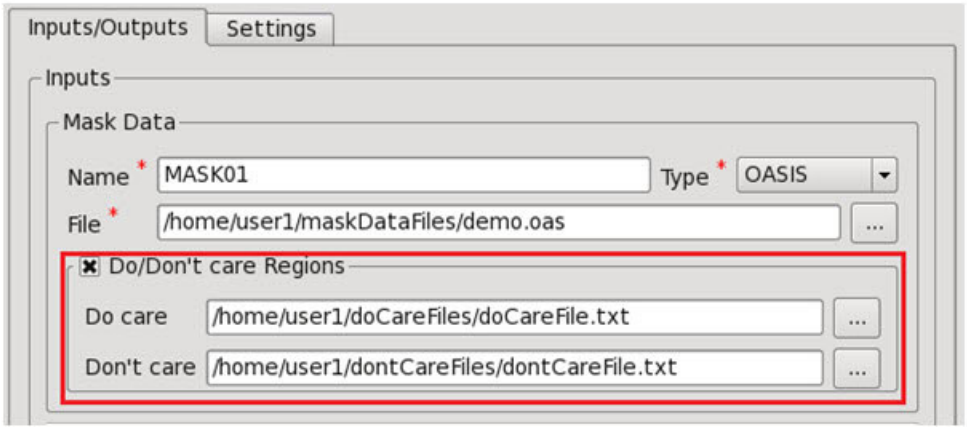
Figure 2-8. Layout Name

   - Do-care and don't-care regions

i. To specify do-care and don't-care regions (areas of the layout to either focus on or ignore), click the **Do/Don't care Regions** check box.

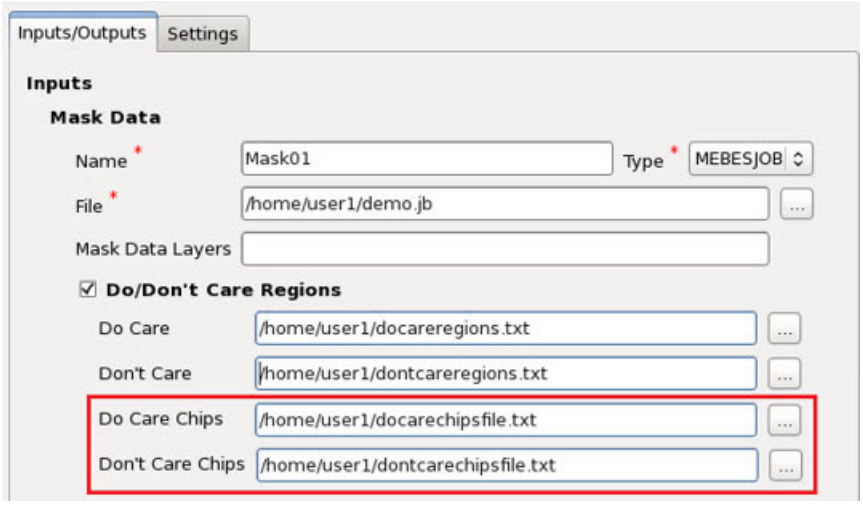
ii. Select do-care and don't-care region files using the **Browse** (**...**) button or specify the path in the corresponding input boxes. For details on do-care and don't-care region files, refer to the do_care_region_file and do_not_care_region_file descriptions in "Parameters File Format" on page 87.

Figure 2-9. Do/Don't Care Regions



iii. For MEBES JOBDECK, in addition to do-care and don't-care region files, you can specify job deck chip names to define the do-care and don't-care regions. Select do-care and don't-care region files using the **Browse** (**...**) button or specify the path in the corresponding input boxes. For details on do-care and don't-care region files, refer to the do_care_jdchips_file and do_not_care_jdchips_file descriptions in "Parameters File Format" on page 87.

Figure 2-10. Do/Don't Care Chip File Names



- Blank Mask Inspection File(s)

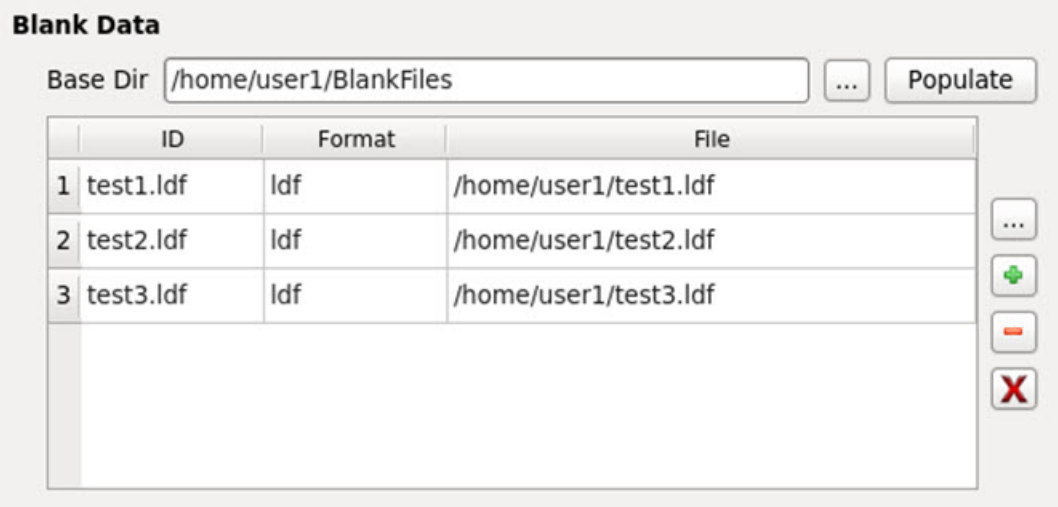
- To select blank mask inspection files, specify the base directory and click **Populate**. All blank mask files located in the directory specified in the Base Dir field are selected. If there are duplicate blank files (the files with same ID or base name), the following dialog box appears:

Figure 2-11. Duplicate Blank Data Entries Detected

If you click **Skip**, the older blank files are retained and the duplicate files from the new Base Dir are ignored. If you click **Overwrite**, the older duplicate blank files are removed and files from a new Base Dir are added instead. Click **Cancel** to cancel the operation.

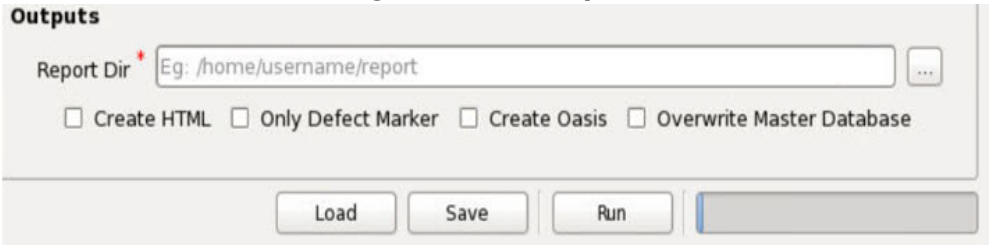
- To add a new row in the table, click the green plus (+) button. Enter the ID, Format and File information in the entry fields of the new row. You can also click the **Browse** (**...**) button to locate a blank mask file.
- To delete a row from a table, select the row and then click the red minus (-) button. To delete all selected blank mask files click red **X** button.

Figure 2-12. Blank Data



5. In the Outputs area of the **Inputs/Outputs** tab, specify outputs.

Figure 2-13. Outputs



- Specify the location where reports are generated in the Report Dir entry field (or click the **Browse** (**...**) button to specify the location).
- To overwrite the database result of previous Design-Blank(s) in the current run, click **Overwrite**.
- To generate an HTML report, click **Create HTML**.
- If **Only HTML Marker** is checked, only the defect marker is added to the visualization RDB file. If unchecked, both the defect and margin markers are added to the visualization RDB file.
- To generate an OASIS report, click **Create OASIS**.
- To show the results of a previous run, click **View Last Report**.

Figure 2-14. View Last Report

Note

The **View Last Report** button only appears if you already ran Calibre MDPDefectAvoidance and you have a valid result file.

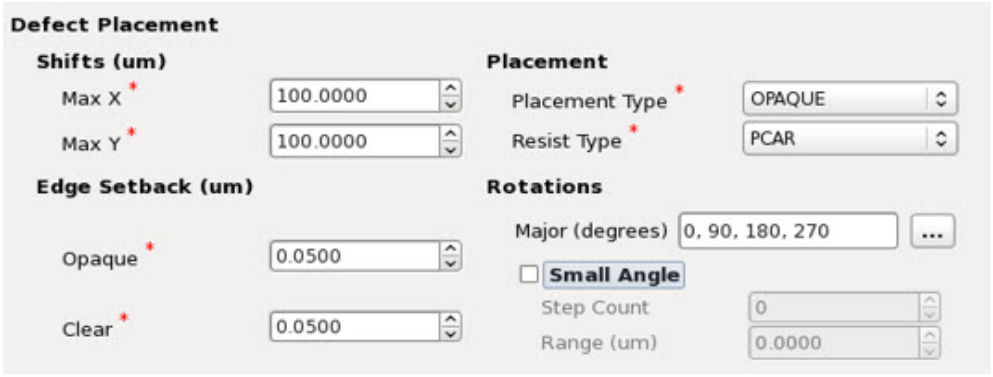
Specifying Defect Avoidance Run Settings

Use the Calibre MDPDefectAvoidance dialog box to specify defect avoidance run settings, including defect placement, defect priorities, mask data layer rules (for layer priorities), and filtering out defects by size, area, or types.

Procedure

1. In the Calibre MDPDefectAvoidance dialog box, click the **Settings** tab.
2. In the **Settings** tab, you can set values for the following parameters:
   - Defect Placement

Figure 2-15. Defect Placement



   - **Shifts** — Specify the Max X and Max Y shift values, the furthest distance the pattern can be translated relative to the blank in X- and Y-dimensions. For details, refer to the max_shift_x and max_shift_y descriptions in “Parameters File Format” on page 87.
   - **Placement** — Select the Placement Type (the region where defects should be placed such that it does not affect the patterns) and Resist Type (the image tone type). For details, refer to the defect_placement_type and resist_type descriptions in “Parameters File Format” on page 87.
   - **Edge Setback** — Specify the Edge Setback values, the distance from the edge of the drawn shape to the edge of the defect. For details, refer to the clear_edge_setback and opaque_edge_setback descriptions in “Parameters File Format” on page 87.
   - **Rotations** — Specify major and minor angle rotations.
     - **Major (degrees)** — Valid values are All, 0, 90, 180 and 270. You can specify

combinations of all four angles. For example, you can specify “180” or “0, 90, 180”.

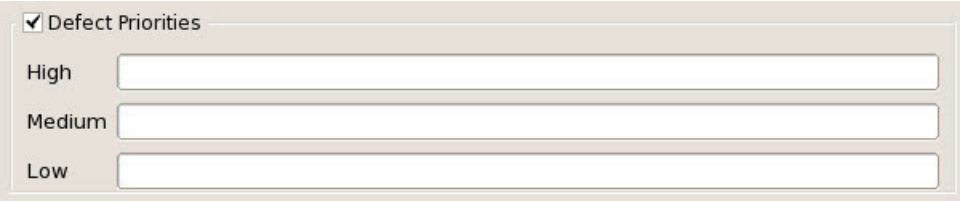
Select angles by clicking the **...** button. Enter “All” to consider all four angles. If no angle is specified, all four angle are considered by default.

- **Small Angle** — Click the check box to specify a small angle value, the length of the arc subtended by two radii after a micro rotation. The Range (um) value cannot be greater than the minimum values displayed in the Shifts (um) field.

  For example, if Max X is 50 microns and Max Y is 40 microns, then the maximum value for Range (um) is 40 microns. For details, refer to the small_angle_rotation_range and small_angle_step_count descriptions in “Parameters File Format” on page 87

- Defect Priorities

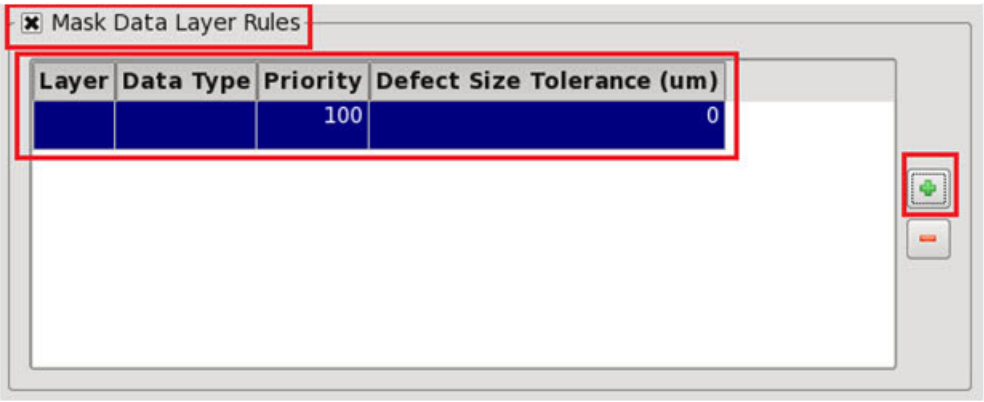
Figure 2-16. Defect Priorities



To enable Defect Priorities (the assignment of which defect is more critical than another), click the check box (by default, it is unchecked). Use this section to optionally specify defect class codes. For details, refer to the defect_type description in “Parameters File Format” on page 87.

- Mask Data Layer Rules

Figure 2-17. Mask Data Layer Rules



This table is required for layer priorities operations. For details, refer to the layer priorities keywords as described in “Parameters File Format” on page 87.

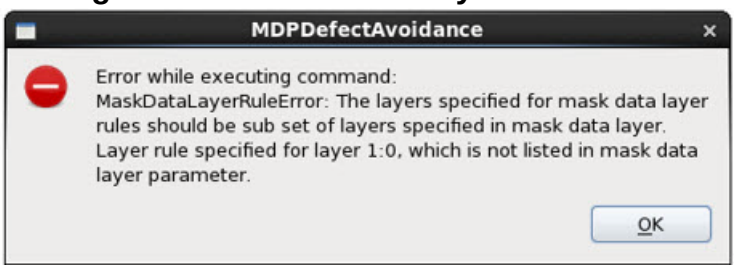
To enable this section, click the Mask Data Layer Rules check box, then click the green plus (+) button to add a new rule.

- **Priority and Defect Size Tolerance (um)** — By default the Priority value is 100, and the Defect Size Tolerance (um) value is 0. However you can double-click each value to modify them.
- **Layer** — In this table, the Layer column is required. Double-click the Layer value, and enter the layer number. If Layer is not specified, an error message is generated.
- **Data Type** — You can specify a Data Type for a layer by double clicking the Data Type value. This should be specified only for OASIS design formats.

To delete a row from this table, select the row first and then click the red minus (-) button.

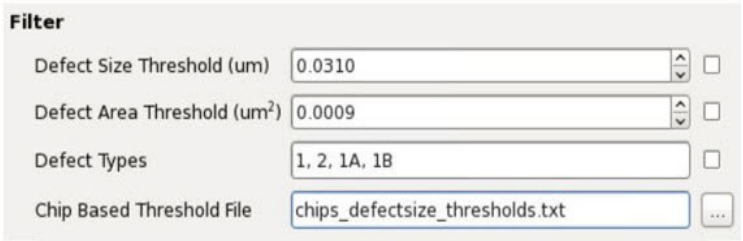
If layers specified in the rule are not a subset of the layers specified in the mask data layer input, the following error is issued:

Figure 2-18. Mask Data Layer Rule Error



- Filter

Figure 2-19. Filters



You can filter out defects based on the Defect Size Threshold, Defect Area Threshold, or Defect Types values. Filtered defects are not considered while avoiding the defects in the tool.

In addition, you can filter out data from some or all of the job deck chips if the size of the defect is less than the defect size threshold (as specified for a chip in the chip-based threshold file). For details on filters, refer to the descriptions of global_defect_size_threshold, global_defect_area_threshold, global_defect_type_filters, and jdchips_thresholds_file in “Parameters File Format” on page 87.

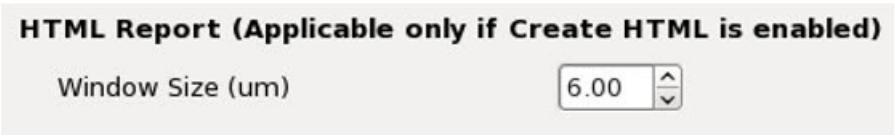
- HTML Report

  The section contains settings used for HTML report generation and is applicable only if **Create HTML** in the Outputs section is selected.

  - Window Size (um)

    This setting is used to define the clip size for the HTML Report. The default window size is 6 microns, 3 microns on each side from the defect location.

Figure 2-20. Window Size in HTML Report



    The layout clip in the HTML Report also considers defect size for clipping so that its window size is either equal to or more than the value specified in the **Window Size** entry. For example, if the window size is specified as 6 microns and the defect width and height are 2 x 2 microns, then the window clip in the HTML report is 8 x 8 microns.

- Remote Information

  Select a remote machine information file using the **Browse** button or specify the path in the corresponding input boxes.

Figure 2-21. Remote Information

  The remote machines specified in the file are used for parallel execution of the following jobs using Calibre MDPview:

  - Parallel indexing and binning of OASIS and MEBES chips are available in the MEBES job deck when MEBESJOB is specified for the Mask Data format.

    When reading the MEBESJOB deck, if index and bin (for OASIS) files for OASIS and MEBES chips contained in the MEBESJOB deck are not available, indexing and injection (or binning) of those chips starts in parallel using the supplied remote machines.

  - Parallel defect avoidance runs of multiple blank inspection reports are supplied through a parameter file with the following:

    - The parallel run of multiple blanks is currently supported for JOBDECK format only.

    - For each blank, additional outputs are generated inside the main output direc-

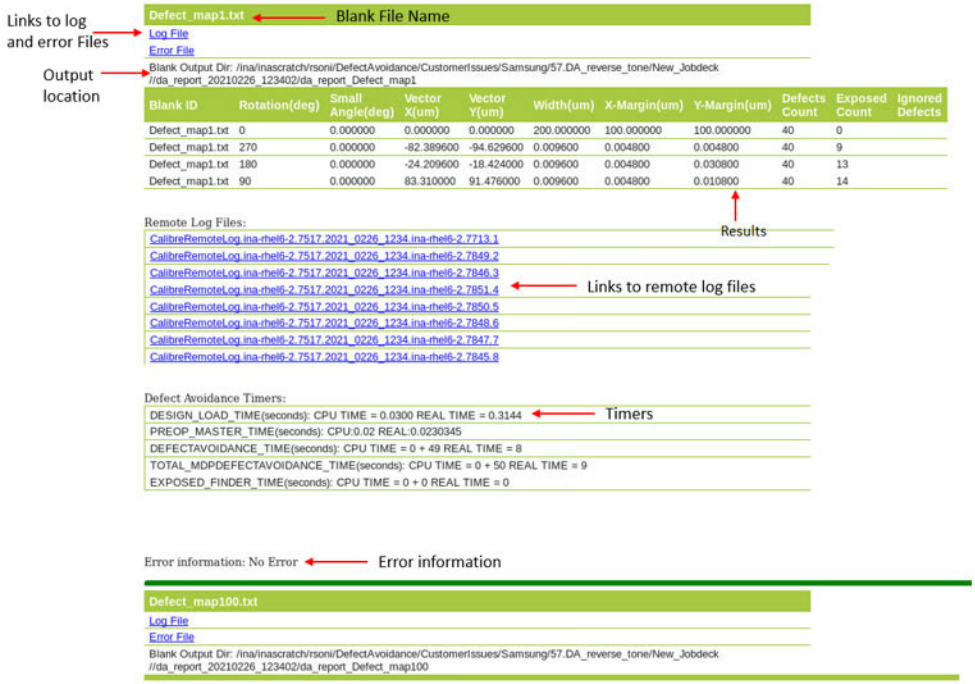
tory specified in the parameter file:

- *param_<blankid>.txt* — Parameter file for each blank inspection report.
- *<blankid>_<hostname>.[log/error]* — Log and error file for each blank inspection report.
- *output_dir/da_report_<blankid>/da_report_<timestamp>* — Output report directory for each blank inspection report.

- Defect Avoidance Parallel Run Report

  When blanks are executed using distributed MT or MTflex mode (-remotemachinesfile <*filename*>), an HTML report named *parallel_run_report.htm* is generated in the output directory. The report displays information as shown in the following figure.

Figure 2-22. Report Generated From a Distributed MTflex Run



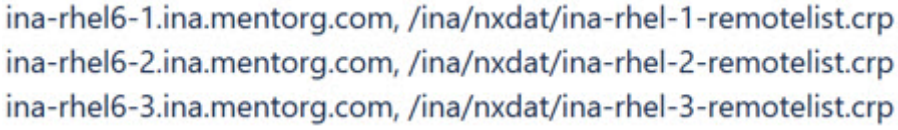
- Parallel HTML report generation

  For information, refer to "Generating an HTML Report in Parallel Using Remotes" on page 76.

The format of the remote machines file contains a remote machine name and an optional Calibre Remote Protocol (CRP) used for a Calibre MTflex file separated by commas in each line:

```
<hostname>, <absolute path of CRP file>
```

Figure 2-23. Example Remote Machines File



The following also applies:

- Siemens Digital Industries Software recommends that index and bin file generation be executed as a separate step prior to executing a defect avoidance run. This can be achieved by using following command:

  ```
  calibremdpv -a da avoid parameter_file  [-remotemachinesfile
  <remotemachines_filename> [-skipda] ]
  ```

  The -skipda keyword stops execution once the index and bin fields are generated.

- During parallel defect avoidance execution, Calibre MDPDefectAvodiance is run in Calibre MTflex mode on the CRP files specified; otherwise, Calibre MDPDefectAvoidance runs in MT mode.

- For parallel indexing and binning, the chip files placed inside the JOBDECK should be accessible from hosts specified in *remotemachinesfile.txt*. For example, a layout file should be readable from host ina-rhel6-1.ina.mentorg.com (shown in the previous figure).

- In the case of MEBES chips placed in the JOBDECK, MEBES indexing is supported only on MT mode. CRP files in *remotemachinesfile.txt* are not required.

- In the case of OASIS chips placed in the JOBDECK:

  - If the CRP file is not specified in *remotemachinefile.txt*, indexing and injection or binning is executed in MT mode on the designated remote machine.

  - The default bin size is 250 um. The minimum bin size is 10 microns. You can generate bin files of a desired bin size by setting the following environment variable:

    ```
    setenv CALIBRE_MDPDA_OASIS_BINSIZE 50
    ```

- To store the index more compactly, use the following environment variable:

  ```
  setenv CALIBRE_PLACEMENT_REPETITION_ENABLE 2112018
  ```

  This addresses memory issues encountered while indexing OASIS files containing many repetitions of the same cell. Indexing time increases, but memory usage and index file size decrease significantly.

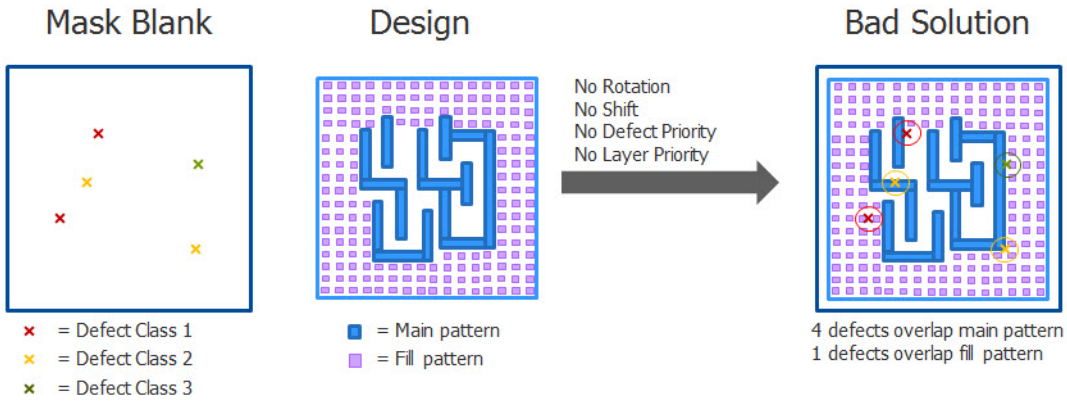
- During the parallel distributed execution, environment variables of the primary machine's process are supplied to the remote machines. The following environment variables are output in the remote logs:
  - MGC_HOME
  - HOME
  - CALBRE_MDPDA_OASIS_BINSIZE
  - CALIBRE_PLACEMENT_REPETITION_ENABLE
  - MDP_VBOASIS_INJECTION
  - MDP_VBOASIS_INJECTION_SIZE
  - MDP_VBOASIS_INJECTION_PRESERVE
  - CALIBRE_DA_HTML_REPORT_GENERATION_TIMEOUT

SSH is used for logins to remote machines in parallel indexing. For more information, refer to "Setting Up a Connection for Remote Machine Information" on page 39.

Examples

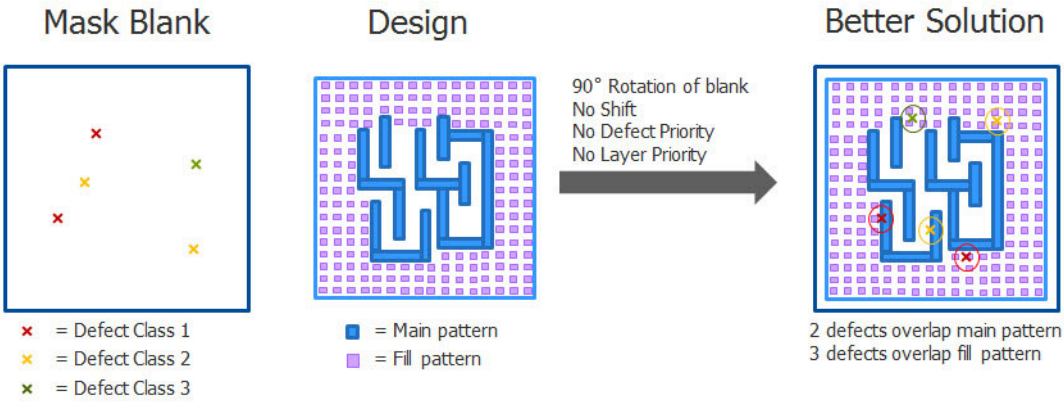
The following examples illustrate the effects on mask blanks and patterns when applying defect avoidance run settings, specifically rotation, shifts, and defect and layer priorities. In the first example, there is no settings applied, thus, there is rotation, shift, or priorities to affect the blank and layout.

Figure 2-24. No Rotation, No Shift, and No Defect Priorities



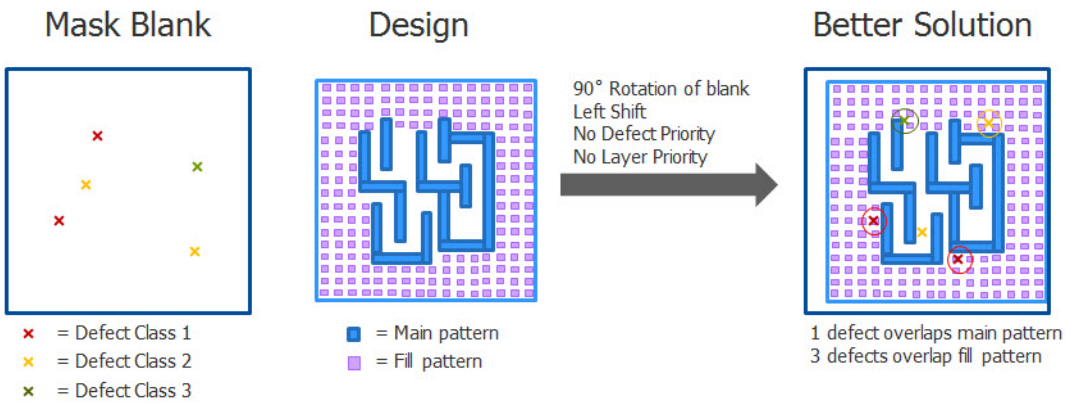
In the next example, the mask blank is rotated 90-degrees only.

Figure 2-25. 90-Degree Rotation of Blank



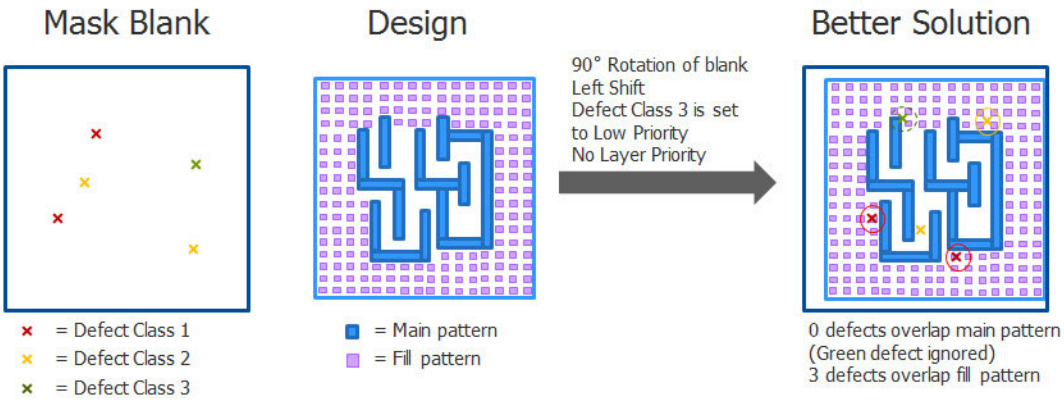
In this example, the blank is both rotated 90-degrees, and a left-shift is applied.

Figure 2-26. 90-Degree Rotation of Blank + Left Shift



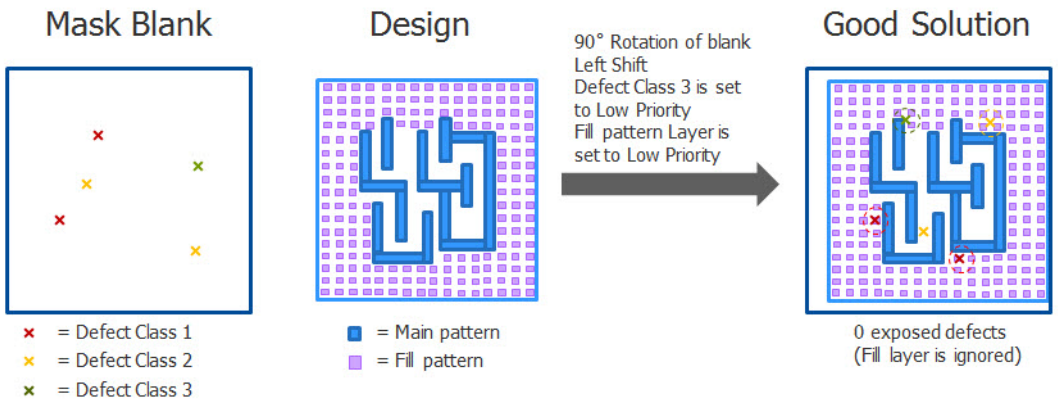
In this example, the blank is rotated 90-degrees, a left-shift is applied, and a defect priority (Defect Class 3 is set to Low) is specified.

Figure 2-27. 90-degree Rotation of Blank + Left Shift + Defect Priority



In this example, the blank is rotated 90-degrees, a left-shift is applied, a defect priority (Defect Class 3 is set to Low) is specified as well as a layer priority (the fill pattern layer is set to Low).

Figure 2-28. 90-Degree Rotation of Blank + Left Shift + Detect Priority + Layer Priority



Setting Up a Connection for Remote Machine Information

The Remote Information section of the Calibre MDPDefectAvoidance interface allows you to specify a file containing information on remote machines. When the MEBES job deck containing no OASIS or MEBES chip index files is read, indexing for those chips starts in parallel using the remote machines specified in the information file. For this process, remote machines can be logged into using SSH.

Prerequisites

- The location of the SSH executable (such as */user/bin*) should be added to PATH environment variable.

Procedure

To set up an SSH connection:

a. Generate public and private keys from the primary machine by entering the following command on the primary machine:

```
ssh-keygen
```

If you want to keep the default names, press the Enter key three times.

This command generates an *id_rsh.pub* file inside your */home/$USER/.ssh* directory. The identification is saved in */home/$USER/.ssh/id_rsa*. The public key is saved in the */home/$USER/.ssh/id_rsa.pub* location.

b. Copy the public key, the *id_rsh.pub* file, and rename it to *authorized_keys*. Place the copy of your */home/$USER/.ssh* directory on the remote machines.

c. You can change the permission of the *authorized_keys* file using the following command:

```
chmod 600 /home/$USER/.ssh/authorized_keys
```

d. Perform an SSH login onto the remote host. To test for ssh command execution (without a password) on the remote host, run one of the following commands from a primary machine:

```
ssh –n –o BatchMode=”yes” <username>@<remotehostname> ls
```

OR

```
ssh -l <username> <remotehostname> ls
```

The second command executes an ls command on the remote machine and lists file names.

e. If issues persist, copy the *id_rsh.pub* and *authorized_keys* files to the remote machine’s */home/$USER/.ssh* directory using an ssh-copy-id command:

```
ssh-copy-id <username>@<remotehostname>
```

The HOME and MGC_HOME environment variables should be supplied to any remotes set in the primary host machine.

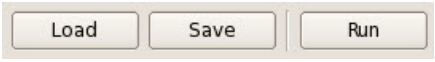
Saving and Loading a Parameter File

You can export all Calibre MDPDefectAvoidance parameter values into a parameter file.

Procedure

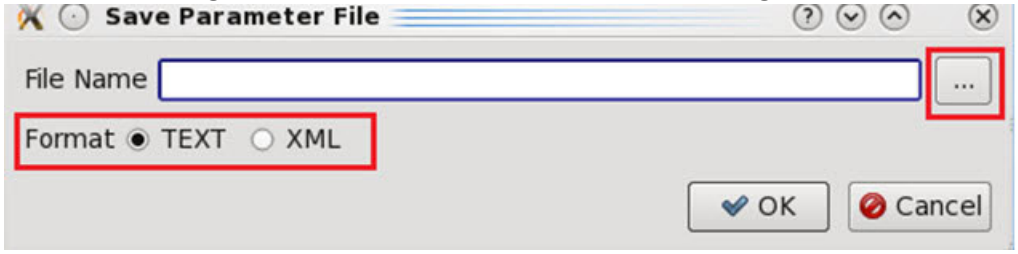
1. In the Calibre MDPDefectAvoidance dialog box, click **Save** at the bottom of the interface.

Figure 2-29. Load and Save Buttons



The Save Parameter File dialog appears.

Figure 2-30. Save Parameter File Dialog Box



2. Specify the file name and path in the entry field, or click the **Browse** (...) button to locate a path.
3. Select the parameter file format: TEXT or XML.
4. Click **OK**.
5. To load a saved parameter file, click the **Load** button on the Calibre MDPDefectAvoidance dialog box.

   If you attempt to load a parameter file that contained either a database file path or a report directory path that was invalid or missing write permissions, a warning message appears, but gives the option to either continue loading or abort the run. If you click **Continue**:

   - The invalid Settings parameters are ignored and previous settings values are retained for invalid ones. Valid Settings parameters are loaded to the GUI.
   - The invalid paths are loaded to the GUI and the errors are highlighted in red.

   For more information on parameter files refer to "Parameters File Format" on page 87.

Using Calibre RVE for Output Visualization

You can visualize the result of Calibre MDPDefectAvoidance by plotting the transformed defects coordinates using Calibre RVE.

Prerequisites

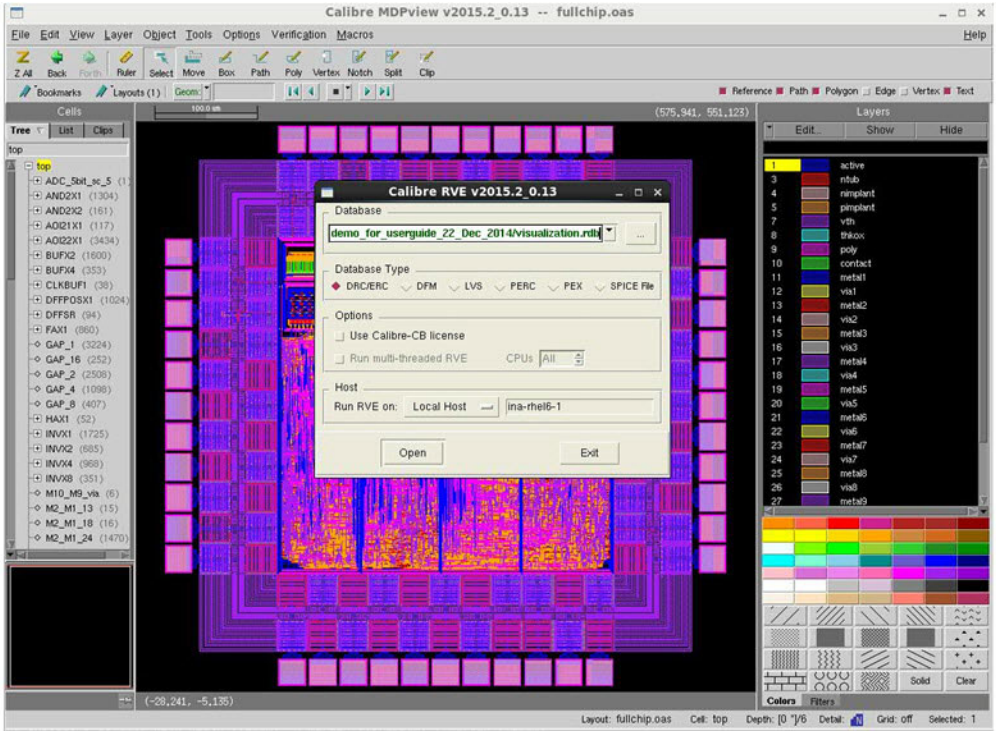
- A valid license for Calibre MDPview and Calibre RVE.
- A layout is loaded into Calibre MDPview.
- Calibre MDPDefectAvoidance has made a successful defect avoidance run, creating a valid visualization results database (*visualization.rdb*). See "Visualization RDB Database File" on page 66.

Procedure

1. Launch Calibre MDPview and open a design file. For more information, refer to the *Calibre MDPview User's and Reference Manual*.

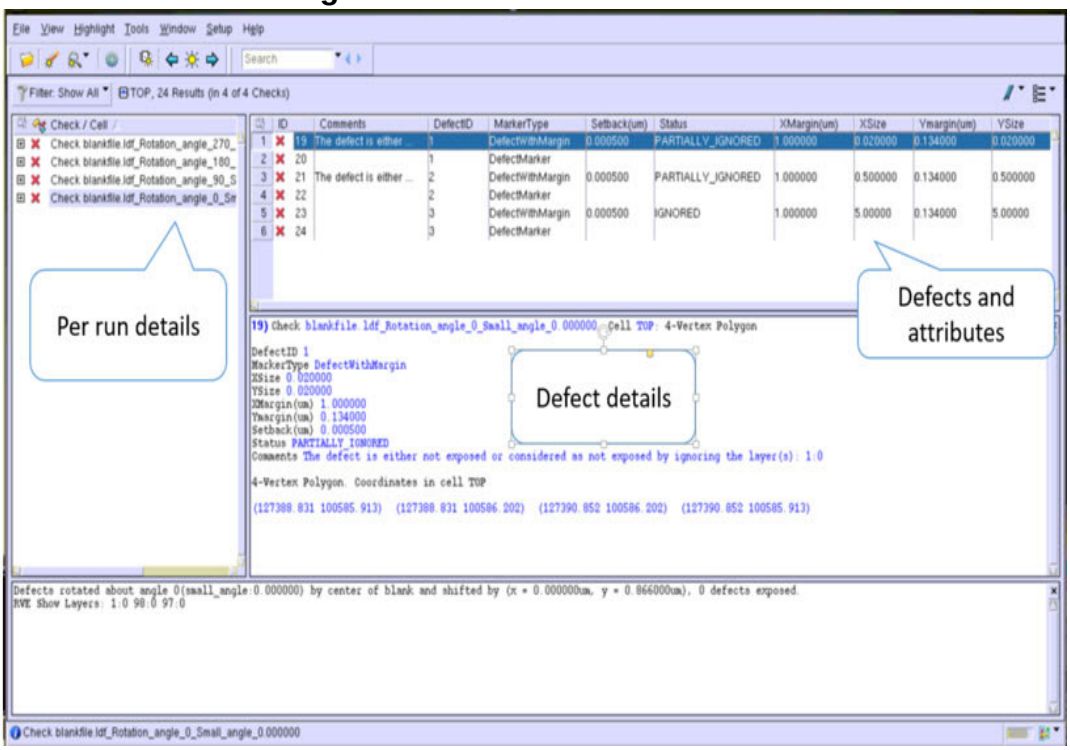
2. In Calibre MDPview, select **Verification > Start RVE** to launch Calibre RVE. The Calibre RVE window appears.

Figure 2-31. Calibre RVE



3. In the Calibre RVE dialog box, provide a valid visualization results database file (located inside a unique directory created inside the one specified by the report_directory or ReportDirectory parameter) in the Database field and click **Open**. A Calibre RVE viewer window appears.

Figure 2-32. Calibre RVE Viewer



The left side pane contains details of each run. The right top pane displays all the defects, which are rotated and shifted by a solution vector. The right bottom pane displays the details of the defect.

There can be two types of markers, DefectMarker and DefectWithMargin. DefectMarker represents only the defect window. DefectWithMargin is a bounding box created by considering the defect, user-supplied setback, and margin reported in the output *result.txt* file. By default, DefectMarker followed by DefectWithMargin are available in the visualization RDB file. If the only_defect_marker parameter is set to true in the Calibre MDPDefectAvodiance parameter file, only DefectMarker is available in the RDB file.

4. Double-click on any defect to plot the defect location on the design layout.

Figure 2-33. Plotting a Defect in the Layout

Use the Calibre RVE interface to navigate through all the defects using RVE and verify the results generated by Calibre MDPDefectAvoidance.

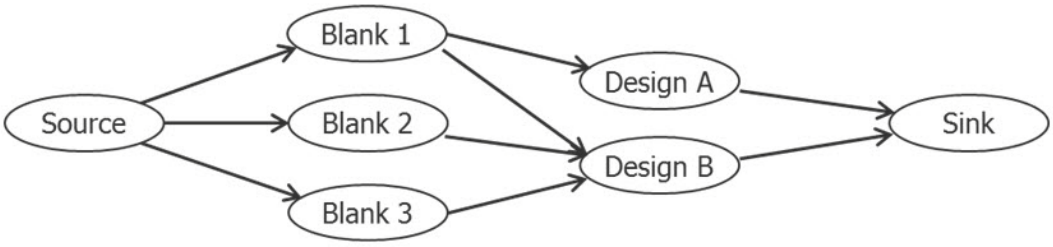
For more details on how to use Calibre RVE, refer to the *Calibre RVE User's Manual*.

Using the Blank-Design Pairing Optimizer

The Blank-Design Pairing Optimizer maximizes the number of manufacturable masks given the blanks and designs. Once the best solutions for individual blank-design pairings have been determined using defect avoidance, optimization across all blanks and designs can be performed using the Blank-Design Pairing Optimizer. The optimizer uses the width of the solution and exposed defects count as cost factors to compute the optimal solution.

Figure 2-34 illustrates an example with three blanks and two designs available for manufacturing.

Figure 2-34. Three Blanks and Two Designs Optimization Problem



In this scenario, Design A can be manufactured using Blank 1 and Design B can be manufactured using Blank 1, Blank 2 or Blank 3. However, using Blank 1 for Design B is not the optimal solution because that leaves Design A without any suitable blank. The optimizer makes sure that the solution found has a "flow" from both Design A and Design B to the Sink node.

Prerequisites

- Calibre MDPDefectAvoidance has successfully made a defect avoidance run.
- A valid Blank-Design database must be selected and loaded into Calibre MDPDefectAvoidance (see "Blank-Design Relational Database" on page 63).

Procedure

1. Use one of the following methods to invoke the pairing optimizer:

   Using the Calibre MDPDefectAvoidance GUI:

   a. Invoke Calibre MDPDefectAvoidance from Calibre MDPview (see "Calibre MDPDefectAvoidance Modes of Operation" on page 17).

   b. In the Calibre MDPDefectAvoidance dialog box, click the **Blank-Design Pairing** tab. There are two sub-tabs, **Pairing Optimizer** and **Import/Export**.

   c. Click the **Pairing Optimizer** tab. The **Pairing Optimizer** tab displays the availability status of blanks and designs, as well as the exposed defect counts and usability factor (UF).

   Using batch mode:

   a. In a command shell, use the following invocation syntax:

      calibremdpv -a da optimize {***database_file_name*** | **-paramfile** ***optimizer_parameter_file***} [-outputfile *output_file_name*]

      where:

      - The ***database_file_name*** is the SQLite database that contains the blank, design, and pairing information from the defect avoidance run. The supported extensions are *.db* or *.sqlite* in case-sensitive file names such as *test1.db* or *test2.sqlite*.
      - The **-paramfile** is an XML file that specifies the options that are available in the GUI. Examples include the status (Available, Unavailable, and Used) of a blank or design, whether a blank-design pair should be considered for pairing or not, and so on.

        If the parameter file is not specified, the pairing run uses the current state of the database (depending on the availability of blanks and designs).

        Using the "da optimize" argument, you can either specify the database file name or optimizer parameter name, but not both. The format of the pairing optimizer file is described in "Pairing Optimizer Parameter File Format" on page 109.
      - The -outputfile is an optional keyword that specifies the name of an XML file in which the pairing results are output. If -outputfile is not specified, the results are written to standard output (console).

For example:

```
calibremdpv -a da optimize da.db

calibremdpv -a da optimize -paramfile \
   optimizer_parameter.xml

calibremdpv -a da optimize da.db -outputfile out.xml \

calibremdpv -a da optimize -paramfile \
   optimizer_parameter.xml -outputfile out.xml
```

2. In GUI mode, the **Pairing Optimizer** tab displays the availability status of blanks and designs, as well as the exposed defect counts (EC) and usability factor (UF).

   The usability factor for a blank-design pair determines the usability of the pair. The usability factor ranges between 0 and 1:

   - 1— Specifies that the blank-design pair is the best usable pair.
   - 0 — Specifies that the blank-design pair is not usable.

     Some parameters used to compute the usability factor are configured in the run configuration file, including ConsiderExposedDefectsInUsabilityFactor, MaxSolutionWidthInMicrons, and MaxExposedDefects. For details on configuration parameters, refer to “Run Configuration File Format” on page 106.

   The usability factor is not computed if a pair is marked as excluded or left blank, or design is marked as unavailable or used.

Figure 2-35. Blank-Design Pairing Optimizer

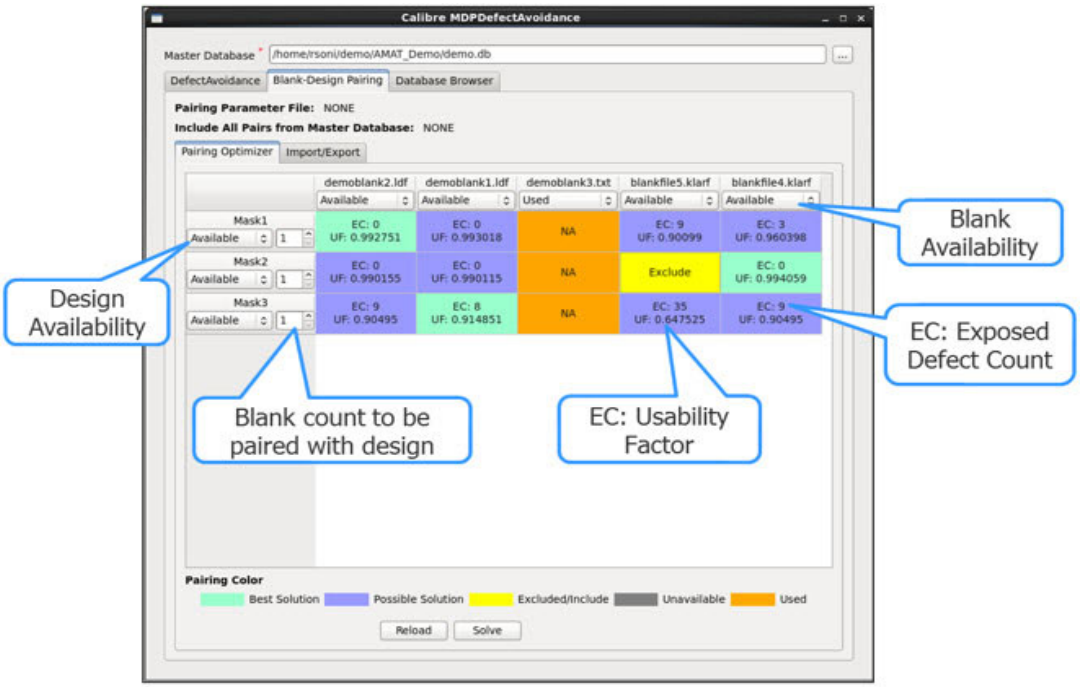


Table 2-1 describes the design availability status flags used in blank-design pairing.

Table 2-1. Blank-Design Pairing Flags

Flag	Description
Available	Blank and Design is available to be considered for pairing.
Unavailable	Blank and Design is not available for pairing.
Used	Blank and Design is already used and cannot be paired with any Design and Blank in the future. If you click **Solve**, the corresponding row and column disappear from window.

3. At any point of time, the Pairing Optimizer displays pairs that are a subset of those available in the Master Database. Click the **Solve** button to invoke the pairing optimizer and click **Reload** to load all the pairs from the Master Database. On clicking **Reload**, the pair states from the Pairing Optimizer are retained.

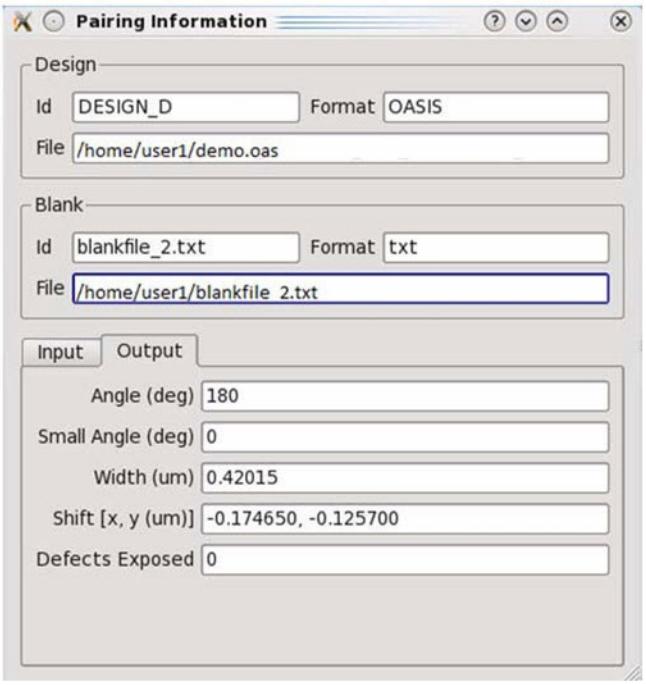
Figure 2-36. Reload and Solve Buttons

The addition and removal of blank, design or pairs in the Master Database is indicated when the **Reload** button is cyan color (as shown in the following figure). When you click **Reload**, the corresponding blank, design, and pairs are loaded to or removed from the pairing optimizer table.

Figure 2-37. Pairing Optimizer Not Synchronized With Master DB

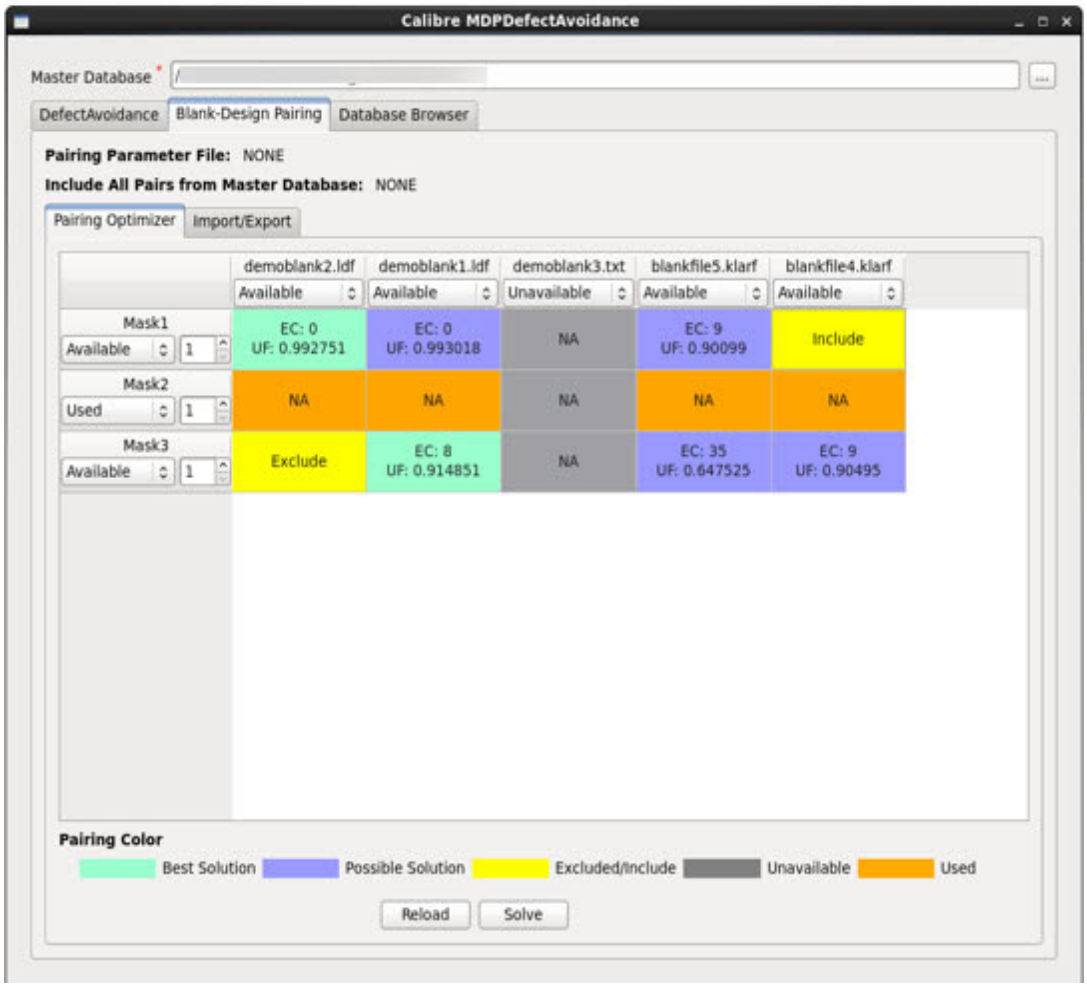
4. To view the information of a particular blank-design pair in the GUI, left-click on a corresponding table cell. The Pairing Information dialog box appears.

Figure 2-38. Pairing Information



5. To exclude or include a blank-design pair for an optimal solution computation, right-click a corresponding table cell. The “Include” or “Exclude” status of the cell indicates whether the blank-design pair is excluded or included, respectively, for an optimal solution computation. If a pair is excluded, the usability factor is not computed for the pair.

Figure 2-39. Exclude or Include a Blank-Design Pair



Results

If you are running the pairing optimizer in batch mode (also if -outputfile has been specified), an output file is generated. The pairing optimizer output contains the information about mask data name, mask blank name, solution status, angle, small angle, x-margin, y-margin, vector-x, vector-y, number of defects exposed, and the usability factor for the blank-design pair.

The status value could be Best, Possible or None. If the status value is None, then other attributes are not written to solution structure as illustrated in the following figure:

Figure 2-40. Pairing Optimizer Example Output

```
<?xml version="1.0" encoding="UTF-8"?>
<dapairing_solutionfile version="1">
   <solution mask_data_name="MASK1" mask_blank_name="demoblank1.ldf">
       <status value="Possible"/>
       <angle value="90"/>
       <small_angle value="0"/>
       <x_margin value="0.2278"/>
       <y_margin value="0.4374"/>
       <width value="0.4556"/>
       <vector_x value="41.6566"/>
       <vector_y value="11.2376"/>
       <exposed value="0"/>
       <usability_factor value="0.990104"/>
   </solution>
   <solution mask_data_name="MASK2" mask_blank_name="demoblank1.ldf">
       <status value="None"/>
   </solution>
   <solution mask_data_name="MASK3" mask_blank_name="demoblank1.ldf">
       <status value="Best"/>
       <angle value="0"/>
       <small_angle value="0"/>
       <x_margin value="10"/>
       <y_margin value="10"/>
       <width value="20"/>
       <vector_x value="0"/>
       <vector_y value="0"/>
       <exposed value="0"/>
       <usability_factor value="0.990297"/>
   </solution>
</dapairing_solutionfile>
```

Import and Export Pairing States

The blank, design, pairs and their states can be exported to a parameter file from the Pairing Optimizer GUI.

Various attributes of the primary database can be exported to the pairing optimizer parameter file, including:

- The reference to the primary database.
- A flag indicating whether to update or not update the primary database
- A flag indicating whether to include or not include all pairs from the primary database.

The saved pairing parameter file can be used to filter the blank-design pairs used for pairing optimization in the GUI (by importing parameter file) as well as in batch mode (by a command line argument).

For specific information on the pairing optimization parameters, refer to “Pairing Optimizer Parameter File Format” on page 109.

Figure 2-41. Import and Export Pairing Parameter File

Importing a Pairing Parameter File

You can import a pairing parameter file using the Pairing Optimizer GUI.

Procedure

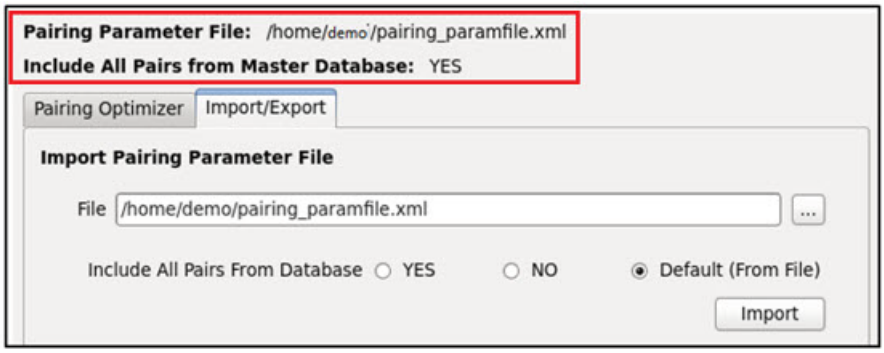
1. In the **Blank-Design Pairing** tab, click the **Import/Export** tab.

2. To import a blank, design, and pairs (and their states) to the Pairing Optimizer GUI, specify the file name in the File entry field under “Import Pairing Parameter File” (shown in Figure 2-42) and click the **Import** button. To browse the parameter file, click the Browse (**...**) button. To overwrite the “Include All Pairs From Database” flag from the parameter file, select either the **YES** or **NO** option.

Figure 2-42. Import Pairing States from a Parameter File

The Pairing Optimizer table is populated with pairs and their states in the pairing optimizer parameter file. The selected parameter file and the “Include All Pairs from Master Database” flag status is displayed on top of the **Pairing Optimizer** and **Import/Export** tabs (see Figure 2-43)..

Figure 2-43. After Import of the Pairing Parameter File



Exporting a Pairing Parameter File

You can export a blank, design, and pairs to a pairing parameter file through the Pairing Optimizer GUI.

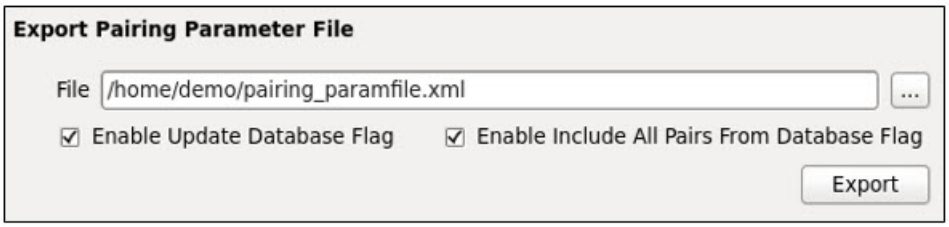
Procedure

1. In the **Blank-Design Pairing** tab, click the **Import/Export** tab.
2. To export a blank, design, and pairs (and their states) to a pairing parameter file, specify the file name in the File entry field under “Export Pairing Parameter File” (shown in Figure 2-44) and click the **Export** button. To browse the parameter file location, click the Browse (**...**) button.

   To enable database updates (specifying “YES” for the flag to enable updates in the parameters file), click the **Enable Update Database Flag** check box. To include all

pairs from the database (specifying “YES” for the flag to include all pairs in the parameters file), click the **Enable Include All Pairs From Database Flag** check box.

Figure 2-44. Export Pairing States to a Parameter File



The exported parameter file can be used for pairing optimization in the Pairing Optimizer GUI as well as in batch modes.

Using the Database Browser

The Database Browser provides views of blanks, designs, and their current pairings as established in the blank-design relational database.

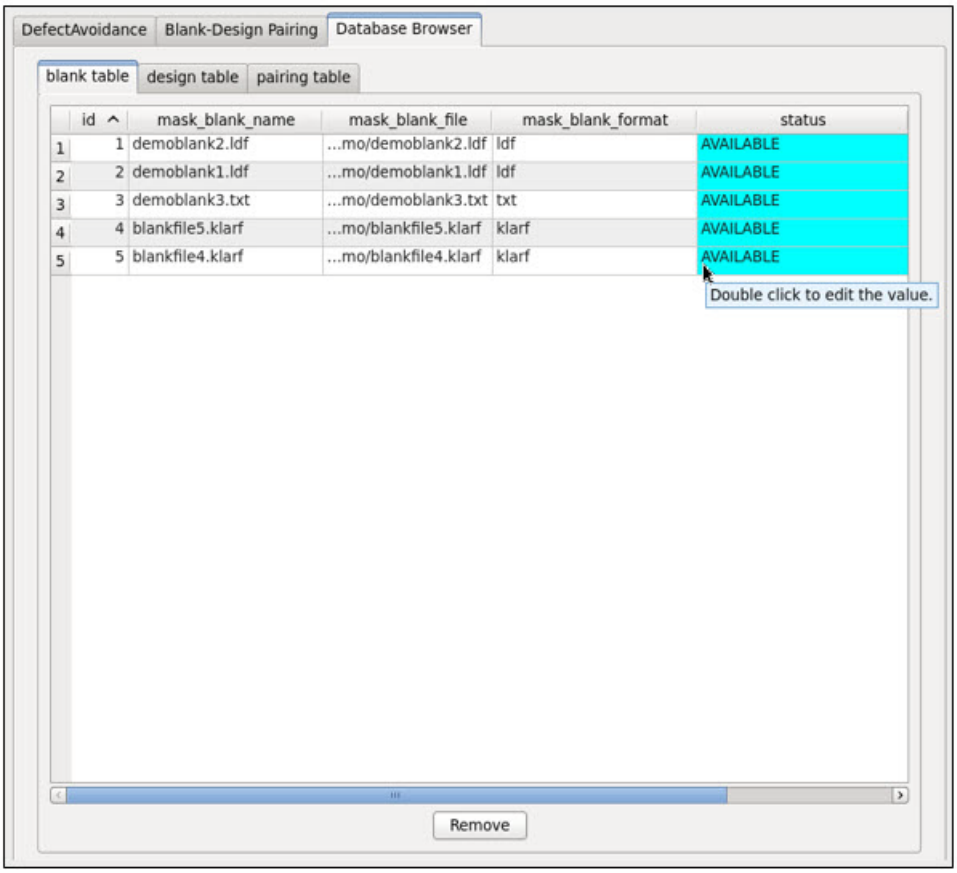
Prerequisites

- Calibre MDPDefectAvoidance has made a successful defect avoidance run.
- A valid Blank-Design database must be selected and loaded into Calibre MDPDefectAvoidance (see “Blank-Design Relational Database” on page 63).

Procedure

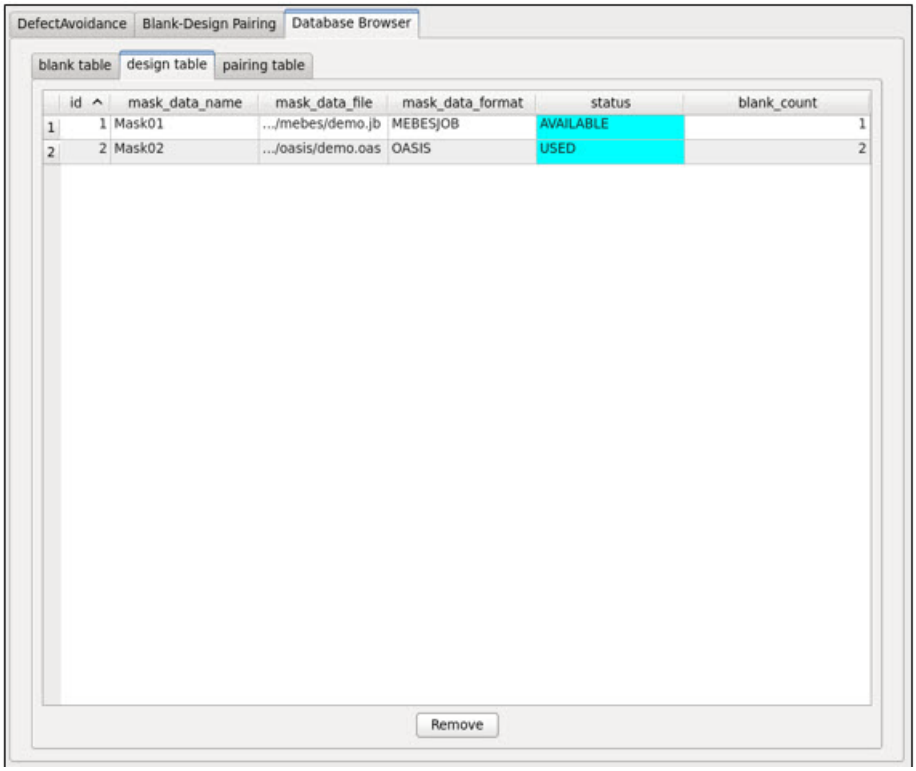
1. Invoke Calibre MDPDefectAvoidance from Calibre MDPview (see “Calibre MDPDefectAvoidance Modes of Operation” on page 17).
2. In the Calibre MDPDefectAvoidance dialog box, click the **Database Browser** tab. There are three sub-tabs, **blank table** , **design table**, and **pairing table**.
   - **blank table** (see Figure 2-45) — Displays a list of blanks from the primary database, including blank name, blank file name, format, and a status column indicating whether the blank is available unavailable or used. As shown in the figure, the status column can be edited after a double-click.

Figure 2-45. Blank Table View



- **design table** (see Figure 2-46) — Displays a list of designs from the primary database, including mask data name, data file name, format, and a status column indicating whether the design is available, unavailable or used. As shown in the figure, the status column can be edited after a double-click.

Figure 2-46. Design Table View



- **Pairing** (see Figure 2-47) — Displays the current blank-design pairings as listed in the primary database.

Figure 2-47. Pairing Table View

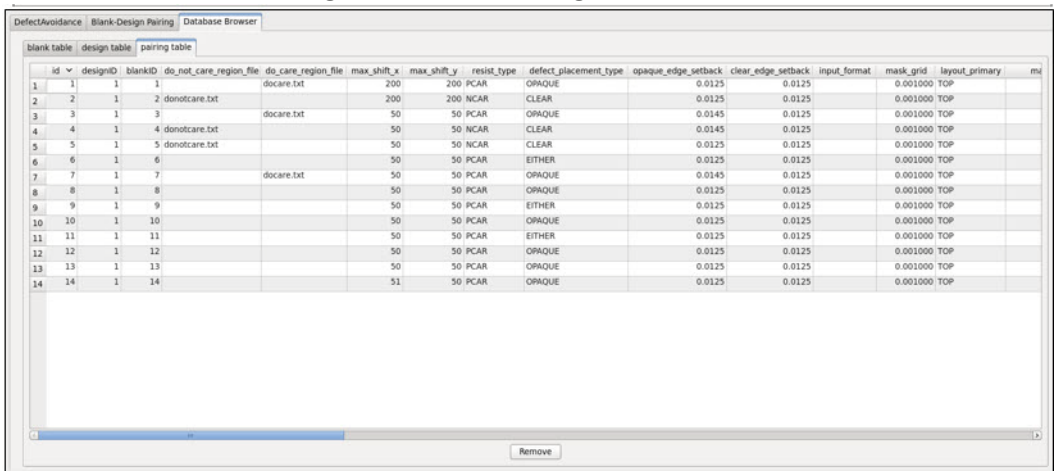


Table 2-2 illustrates the different functionality supported for each Database Browser table.

Table 2-2. Database Browser Functionality

Column	Blank	Design	Pairing	Comments
Sorting	Yes	Yes	Yes	Sorting on all columns.
Column(s) Editing	Yes	Yes	No	Editable columns are represented with a cyan background.
Editable Column Names	status	status	NA	Can be set to Used, Available, or Unavailable.
Row(s) Deletion	Yes	Yes	Yes	

The state of a blank or design can be set in the status column of a corresponding blank or design tables using the Database Browser. If set to Available, the blank is available for pairing. If set to Unavailable, that blank is not available for pairing. If set to Used, the blank is already used and cannot be paired with any design.

For details on blank, design and pairing table columns, refer to "Blank-Design Relational Database" on page 63.

Using the DAVerify Utility

DAVerify is a utility used to automatically cross check the correctness of solution provided by a defect avoidance run. It can be executed post-run and is only supported in command line mode.

Procedure

In a command shell, use the following invocation syntax:

calibremdpv -a da verify *params_file report_dir* [-configfile *filepath*]

where:

- The ***params_file*** is the same parameters file used for a defect avoidance run.
- The ***report_dir*** is the report directory generated by a defect avoidance run.
- The -configfile is an optional keyword that specifies the XML configuration file used for the defect avoidance run.

In the following examples, *params.txt* is parameter file used by a defect avoidance run, *da_report_20200426_100333* is the output directory generated by a defect avoidance run, and *da_config.xml* is configuration file used by a defect avoidance run:

```
$MGC_HOME/bin/calibremdpv -a da verify params.txt \
   da_report_20200426_100333

$MGC_HOME/bin/calibremdpv -a da verify params.txt \
   da_report_20200426_100333 -configfile da_config.xml
```

Results

The result of a DAVerify run is PASS or FAIL status. PASS means the defect avoidance run result is correct and FAIL means that there are issues with the run that might require manual review. The following conditions are applied to decide PASS or FAIL for DAVerify:

- If a defect avoidance run reports a defect as EXPOSED but DAVerify does not report the same, then the result is a FAIL
- If DAVerify reports a defect as EXPOSED but the defect avoidance run does not report the same, then the result is a FAIL.
- In all other cases, the DAVerify result is reported as a PASS.

Note

DAVerify considers only the defect size and setbacks for its query window. Margins reported by a defect avoidance run are not considered.

Defects, a defect avoidance run's status, and DAVerify's status are recorded in a log file *cal_mdpda.log* only when FAIL is reported by DAVerify.

Chapter 3
Calibre MDPDefectAvoidance Formats

Calibre MDPDefectAvoidance determines if shifts are required in layout data to prevent defects from affecting features and then outputs data in various forms in order to analyze the results.

Calibre MDPDefectAvoidance Inputs

A mask or layout data file, blank mask inspection (defect data) files, and control parameter information (in a parameters file) are used as inputs to Calibre MDPDefectAvoidance.

For specific information on the control parameters, refer to “Parameters File Format” on page 87.

Supported Input Formats

Calibre MDPDefectAvoidance supports a number of formats for its input files.

The following mask and layout data formats are supported for Calibre MDPDefectAvoidance:

- OASIS
- VSB
- VSBJOB
- MEBES
- MEBESJOB
- JEOL

The following formats are supported for the layer priorities feature in Calibre MDPDefectAvoidance:

- OASIS
- MEBESJOB

The following blank mask inspection file formats are supported by Calibre MDPDefectAvoidance:

- KLARF 1.2
- Lasertec
- Text

Blank Mask Inspection File

An input for Calibre MDPDefectAvoidance is the blank mask inspection file. This is a text file containing specific details for defects on the blank mask, including location, size, and class ID.

Figure 3-1 illustrates an example of a text blank mask inspection file.

Figure 3-1. Text Blank Mask Inspection File

```
3000.0000 3000.0000 0.01 0.01 Medium 0.03 0.03
3100.0000 3100.0000 0.01 0.01 Medium 0.03 0.03
3200.0000 3200.0000 0.01 0.01 Medium
3300.0000 3300.0000 0.01 0.01 Medium 0.03 0.03
3400.0000 3400.0000 0.01 0.01 Medium 0.03 0.03
3500.0000 3500.0000 0.01 0.01 Medium 0.01 0.01
3600.0000 3600.0000 0.01 0.01 Medium 0.03 0.03
3700.0000 3700.0000 0.01 0.01 Medium 0.03 0.03
3800.0000 3800.0000 0.01 0.01 Medium
3900.0000 3900.0000 0.01 0.01 Medium 0.03 0.03
```

Table 3-1 describes the contents for each column.

Table 3-1. Text Blank Mask Inspection File Columns

Column Number	Description
First	X-coordinate in microns (mandatory double value).
Second	Y-coordinate in microns (mandatory double value).
Third	Defect width in microns (mandatory double value).
Fourth	Defect height in microns (optional double value)
Fifth	Defect class ID (optional text value).
Sixth	Clear setback for the defect. If not supplied, the clear setback specified in the Calibre MDPDefectAvoidance parameter file is considered.
Seventh	Opaque setback for the defect. If not supplied, the opaque setback specified in the Calibre MDPDefectAvoidance parameter file is considered.

Note

Values of the text inspection file are extracted from left to right. For example, the opaque setback cannot be specified without specifying first six columns. The opaque setback is always the seventh column.

Calibre MDPDefectAvoidance Outputs

As a result of a Calibre MDPDefectAvoidance run, several different types of output are generated for analysis.

Text File with Single Run Results

When a defect avoidance analysis is executed, the result is stored in a file named *result.txt* inside a unique directory created under the *report* directory specified. The result file is overwritten in the next run.

Table 3-2 describes attributes included in a result file.

Table 3-2. Results Attributes for a Single Run

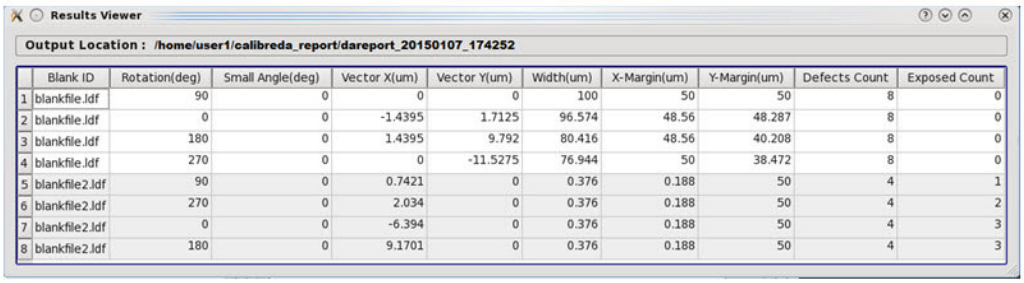
Attribute	Description
Blank ID	ID of blank mask file.
Rotation (deg)	Rotation in angle.
Small Angle (deg)	Small angle rotation where solution has been found.
Vector X (um)	Microns shift in X direction where solution has been reported.
Vector Y (um)	Microns shift in Y direction where solution has been reported.
Width (um)	Defect width, as a double of min of X-Margin and Y-Margin.
X-Margin (um)	The maximum freedom in X-direction by which defects can be shifted without affecting any feature.
Y-Margin (um)	The maximum freedom in Y-direction by which defects can be shifted without affecting any feature.

Table 3-2. Results Attributes for a Single Run (cont.)

Attribute	Description
Defects Count	Number of defects in blank mask file represented by Blank ID.
Exposed Count	Number of defects exposed with the solution reported.
Ignored Defects	Defect types and layers that are ignored to find the solution (applicable for OASIS and MEBESJOB formats). Refer to "Order of Priority for Defects and Layers" on page 102 for details on the priority order of defect types to find the solution for defect avoidance.

When defect avoidance is executed from the GUI, the result is also displayed in a Results Viewer table as shown in Figure 3-2.

Figure 3-2. Results Viewer



	Blank ID	Rotation(deg)	Small Angle(deg)	Vector X(um)	Vector Y(um)	Width(um)	X-Margin(um)	Y-Margin(um)	Defects Count	Exposed Count
1	blankfile.ldf	90	0	0	0	100	50	50	8	0
2	blankfile.ldf	0	0	-1.4395	1.7125	96.574	48.56	48.287	8	0
3	blankfile.ldf	180	0	1.4395	9.792	80.416	48.56	40.208	8	0
4	blankfile.ldf	270	0	0	-11.5275	76.944	50	38.472	8	0
5	blankfile2.ldf	90	0	0.7421	0	0.376	0.188	50	4	1
6	blankfile2.ldf	270	0	2.034	0	0.376	0.188	50	4	2
7	blankfile2.ldf	0	0	-6.394	0	0.376	0.188	50	4	3
8	blankfile2.ldf	180	0	9.1701	0	0.376	0.188	50	4	3

Blank-Design Relational Database

Calibre MDPDefectAvoidance outputs results into a relational database. You can store and manage the best result of each individual mask design and mask blank combination. By using a database, the overall efficiency is improved since it is unnecessary to rerun combinations for results that have already been calculated. The SQLite database is supported by Calibre MDPDefectAvoidance. The supported extensions of SQLite database files are *.db* and *.sqlite* which are case sensitive.

Table 3-3, Table 3-4, and Table 3-5 describe the columns displayed in Blank Mask, Design and Blank-Design Pairing relational database tables.

The Blank-Design Pairing table is used by the Blank-Design Pairing Optimizer. For details, refer to "Using the Blank-Design Pairing Optimizer" on page 44.

Table 3-3. Blank Mask Table Columns

Column Name	Column Type	Description
id	INTEGER	Unique identifier (PRIMARY KEY) for a record in a Blank Mask table.
mask_blank_name	TEXT	Unique identifier for a blank mask file. The base name is used as an identifier for the file.
mask_blank_file	TEXT	Path of a blank mask file with the specified file name.
mask_blank_format	TEXT	Blank inspection file format (for example, LDF, KLARF 1.2, and TEXT).
status	TEXT	Used by the Blank-Design Pairing Optimizer. Valid values are AVAILABLE, UNAVAILABLE, and USED. For more details, refer to “Using the Blank-Design Pairing Optimizer” on page 44.

Table 3-4. Design Table Columns

Column Name	Column Type	Description
id	INTEGER	Unique identifier (PRIMARY KEY) for a record in the Design Table.
mask_data_name	TEXT	Unique identifier for a user-specified design file.
mask_data_file	TEXT	Path of a design file with the specified file name.
mask_data_format	TEXT	Design files type (for example, MEBESJOB, MEBES, OASIS, VSB, VSBJOB, and JEOL).
blank_count	INTEGER	Used by the Blank-Design Pairing Optimizer. Specifies the maximum number of blank masks with which the design should be paired. For more details, refer to “Using the Blank-Design Pairing Optimizer” on page 44.
status	TEXT	Used by the Blank-Design Pairing Optimizer. Valid values are AVAILABLE, UNAVAILABLE, and USED. For more details, refer to “Using the Blank-Design Pairing Optimizer” on page 44.

Table 3-5. Blank-Design Pairing Table

Column Name	Column Type	Description
id	INTEGER	Unique identifier (PRIMARY KEY) for a record in the Blank-Design Pairing Table.
designID	INTEGER	Design ID referring to a design of Design table.
blankID	INTEGER	Unique ID referring to a blank mask in the Blank Mask table.
do_not_care_region_file	TEXT	Reserved for future use.
do_care_region_file	TEXT	Reserved for future use.
max_shift_x	REAL	Maximum shift in X-direction.
max_shift_y	REAL	Maximum shift in Y-direction.
resist_type	TEXT	User-specified information about image tone. Values can be either PCAR or NCAR.
defect_placement_type	TEXT	User-specified information about defect placement type. Values can be either OPAQUE, CLEAR, or EITHER.
opaque_edge_setback	REAL	User-specified minimum distance from the edge of the opaque shape to the edge of the defect.
clear_edge__setback	REAL	User-specified minimum distance from the edge of the clear shape to the edge of the defect.
input_format	TEXT	Type of design file. Valid formats include MEBES, MEBESJOB, OASIS, VSB, VSBJOB and JEOL.
mask_grid	TEXT	The design’s chip units in microns.
layout_primary	TEXT	The design’s top cell.
mask_data_layers	TEXT	Reserved for future use.
angle	REAL	Angle of best solution.
vector_x	REAL	X-shift of best solution.
vector_y	REAL	Y-shift of best solution.
exposed	INTEGER	Number of exposed defects with best solution. 0 if no defect is exposed.
report	TEXT	Path of report directory.

Table 3-5. Blank-Design Pairing Table (cont.)

Column Name	Column Type	Description
is_paired	BOOLEAN	Identifies whether defect avoidance is already executed with a blankID-designID pair.
exclude	BOOLEAN	Used by the Blank-Design Optimizer. If set to 1, the blankID and designID cannot be paired. For more details, refer to "Using the Blank-Design Pairing Optimizer" on page 44.
width	REAL	Width of the "optimal" solution. It is a double of minimum of x_margin and y_margin.
x_margin	REAL	X-margin of "optimal" solution.
y_margin	REAL	Y-margin of "optimal" solution.
small_angle	REAL	Small angle of "optimal" solution if it is found using small angles.

Visualization RDB Database File

After each successful run of Calibre MDPDefectAvoidance, you can view the defect details in a visualization results database file (*visualization.rdb*) that is created inside a unique directory in the reports directory.

The file is created for all defects in a blank file after applying rotations and shift provided by tool on each defect. The file can be loaded using Calibre RVE to review the defect details. For details, refer to "Using Calibre RVE for Output Visualization" on page 41.

Each defect in the visualization results database file contains the attributes described in Table 3-6.

Table 3-6. Visualization Results Database File Attributes

Attribute	Description
DefectID	A unique defect identifier retrieved from the defect record of a blank mask file.
XSize	Defect width in microns.
YSize	Defect height in microns.
Priority	Priority of the defect as specified from the parameter file or Calibre MDPDefectAvoidance dialog box. Refer to the defect_type parameter description in "Parameters File Format" on page 87 for more details.

Table 3-6. Visualization Results Database File Attributes (cont.)

Attribute	Description
Status	Status of a defect. Possible values are as follows: • EXPOSED — The defect is exposed on the high priority layer(s). • NOT_EXPOSED — The defect is not exposed on any layer. • FILTERED — The defect is filtered based on size threshold, area threshold, or defect types. • IGNORED — The defect is ignored to find the solution. • PARTIALLY_IGNORED — The defect is not exposed or considered as not exposed by ignoring certain layers. The Comments attribute provides details about ignored layers. • IN_DONT_CARE_REGION — The defect appears in a don't care region. The defect status values are described in detail in “Defect Status” on page 67.
Comments	Provides details about ignored layers when Status is set PARTIALLY_IGNORED. For example: `The defect is either not exposed or considered as not exposed by ignoring the layer(s): 4:0` This comment states that the defect is ignored on layer 4:0 of the input OASIS file to find the solution.
Defect Coordinates	The defect coordinates in database units after applying rotations and shifts computed by defect avoidance.

Defect Status

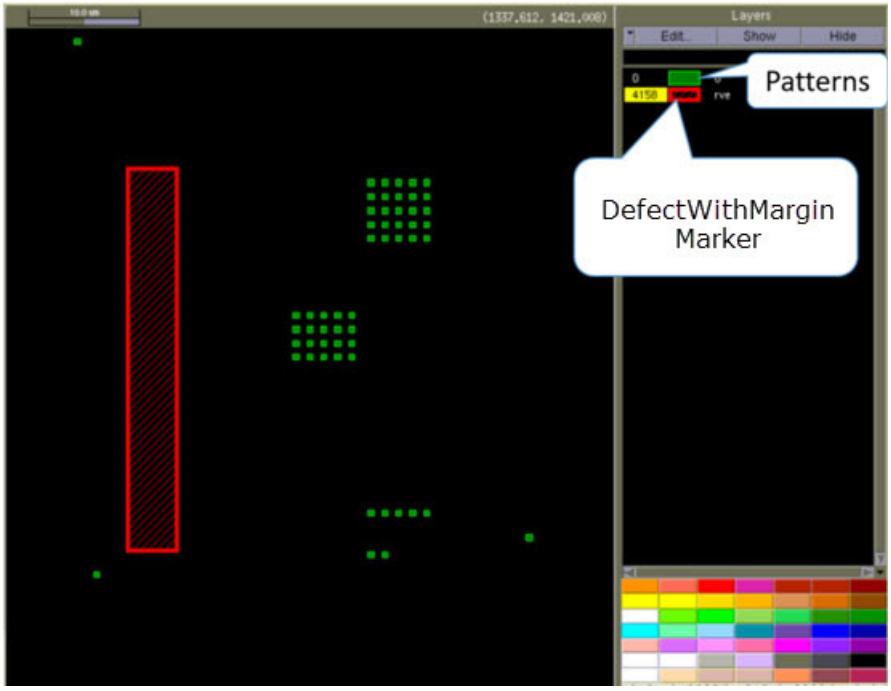
The visualization results database file contains information on the status of a defect. The possible states are:

- EXPOSED — The defect is exposed on the high priority layer(s). In this case, the defect could not be avoided and exposed due to the defect area touching critical patterns. In the following figure illustrating an EXPOSED defect, green shapes are patterns and the red box represents the DefectWithMargin marker (Defect + Setback + Margin) reported in the output. In this example, the marker overlaps the critical patterns and reported as EXPOSED.

Figure 3-3. EXPOSED Defect

- NOT_EXPOSED — The defect is not exposed on any layer. The defect is avoided as it does not touch any pattern. In the following figure illustrating a NOT_EXPOSED defect, the green shapes are patterns and red box is the DefectWithMargin marker (Defect + Setback + Margin) reported in the output. In this example, the marker does not overlap with any of the critical patterns and is reported as NOT_EXPOSED.

Figure 3-4. NOT_EXPOSED Defect



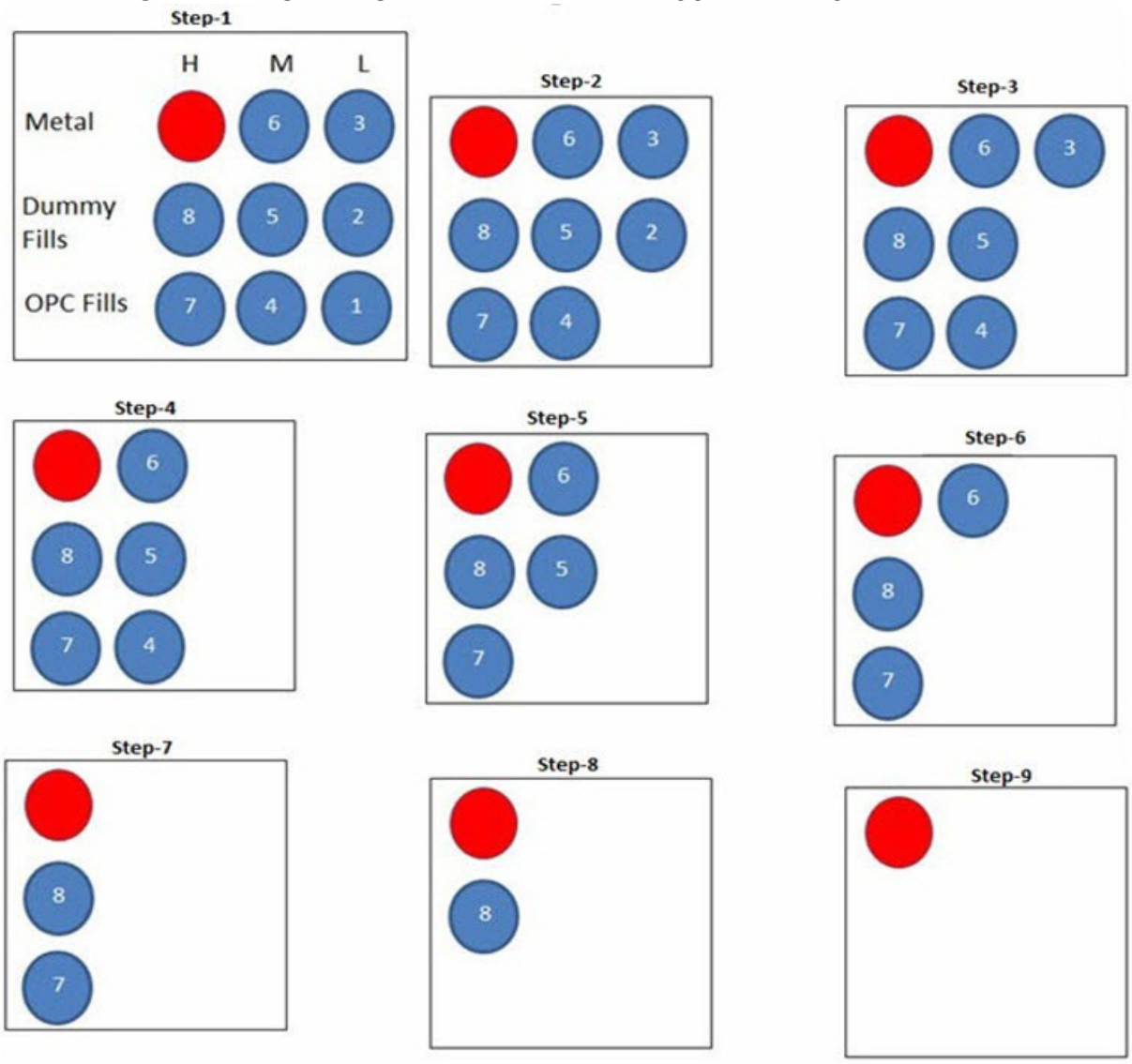
- FILTERED — A defect can be pre-filtered by various conditions such as global defect size threshold, global defect area threshold, and global defect type filters. These parameters are described in “Parameters File Format” on page 87. If a defect satisfies any one of these conditions, it is reported as FILTERED.
- IGNORED — A defect’s status is reported as IGNORED if the solution is found by ignoring certain defect types and the defect is defined as one of the ignored defect types. For example, if there are Low, Medium, and High priority defects and solution is found by ignoring Low and Medium priority defects, then all the Low and Medium priority defect’s status are reported as IGNORED.

  In defect avoidance runs where both defect type and layer priorities are specified, a defect is reported as IGNORED if the defect is defined under ignored defect types that are ignored on all layers to find the solution (in spaces where defects can be placed).

  In the following figure that illustrates ignoring defects based on type and layer priorities, if a solution is found at Step-4 (by ignoring Low priority defect on all the layers) and the defect type is Low, the defect is reported as IGNORED.

  Similarly, if a solution is found at Step-7 by ignoring Low and Medium priority defect on all the layers and the defect type is either Low or Medium, the defect is reported as IGNORED.

Figure 3-5. Ignoring Defects Based on Type and Layer Priorities



Refer to "Order of Priority for Defects and Layers" on page 102 for a description on how solutions are found by ignoring certain defect types and layer priorities.

- PARTIALLY_IGNORED — Not exposed or considered as not exposed by ignoring some layers. The Comments attribute provides details about ignored layers. A defect's status is reported as PARTIALLY_IGNORED if the solution is found by ignoring certain defect types on some of the layers and the defect is defined under one of the ignored defect types.

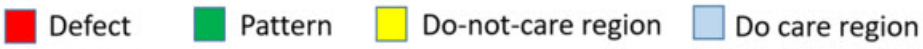
  For example, if a solution is found in Step-2 of Figure 3-5 (by ignoring Low priority defects on OPC Fills) and the defect is defined as Low, then the defect is reported as PARTIALLY_IGNORED. Similarly, if a solution is found at Step-3, Step-5, Step-6, Step-8, and Step-9, then the defect is reported as PARTIALLY_IGNORED if the defect is categorized under any of the ignored defect types.

In addition, a defect is reported as PARTIALLLY_IGNORED if it is ignored on a layer due to the size tolerance defined for that layer. Refer to the description of mask_data_layer_rule in “Parameters File Format” on page 87 for details on how to specify defect size tolerance for a layer.

- IN_DONT_CARE_REGION — A defect is reported as IN_DONT_CARE_REGION if the defect is completely inside user-specified do-not-care regions. The do-not-care regions are defined by the do_care_region_file , do_not_care_region_file, do_care_jdchips_file, and do_not_care_jdchips_file parameters as described in “Parameters File Format” on page 87.

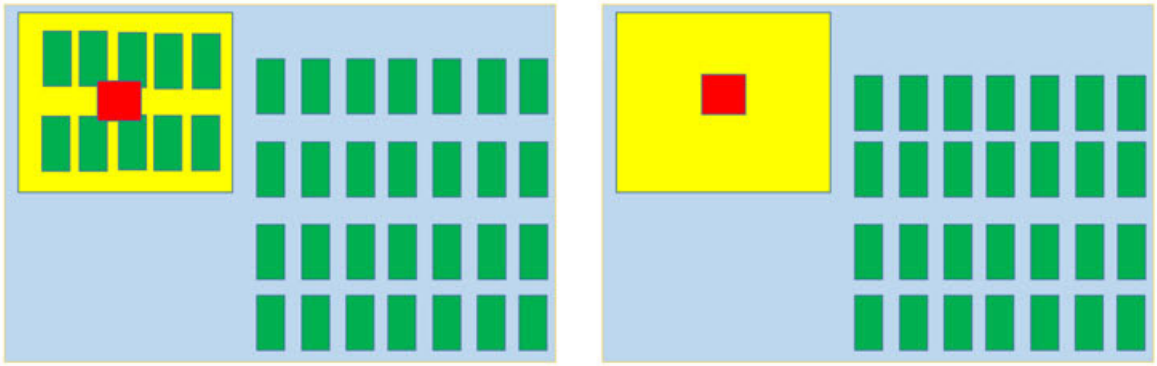
  The following figures illustrate several examples using a different defect status. Each figure uses the following color codes:

Figure 3-6. Color Key for IN_DONT_CARE_REGION Examples



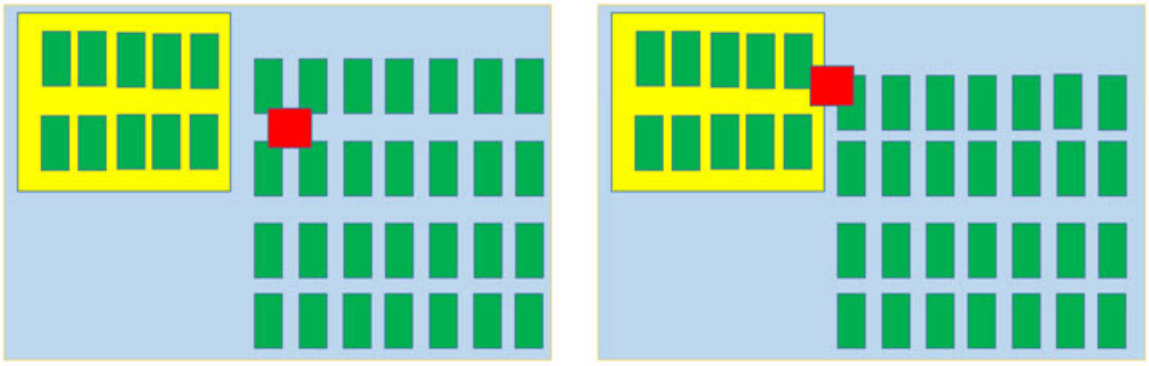
The following figure illustrates IN_DONT_CARE_REGION defects as the defect is completely inside the do-not-care region. It does not matter if the defect overlaps with any pattern or is not inside the do-not-care region.

Figure 3-7. IN_DONT_CARE_REGION



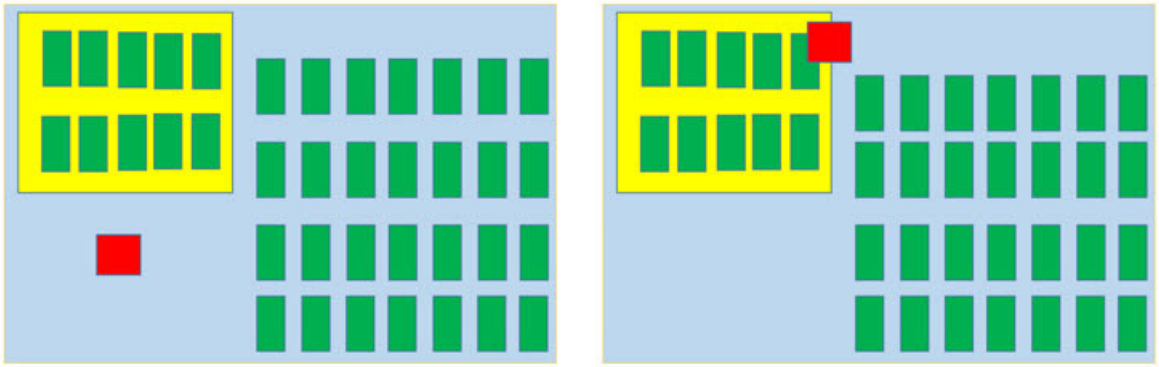
The following figure illustrates EXPOSED defects as the defects are not completely inside the do-not-care regions and touching the patterns available outside the do-not-care region.

Figure 3-8. EXPOSED Defects in Do-Not-Care Regions



The following figure illustrates NOT_EXPOSED defects as the defects are not completely inside the do-not-care regions and do not overlap with patterns outside the do-not-care region.

Figure 3-9. NOT_EXPOSED Defects in Do-Not-Care Regions

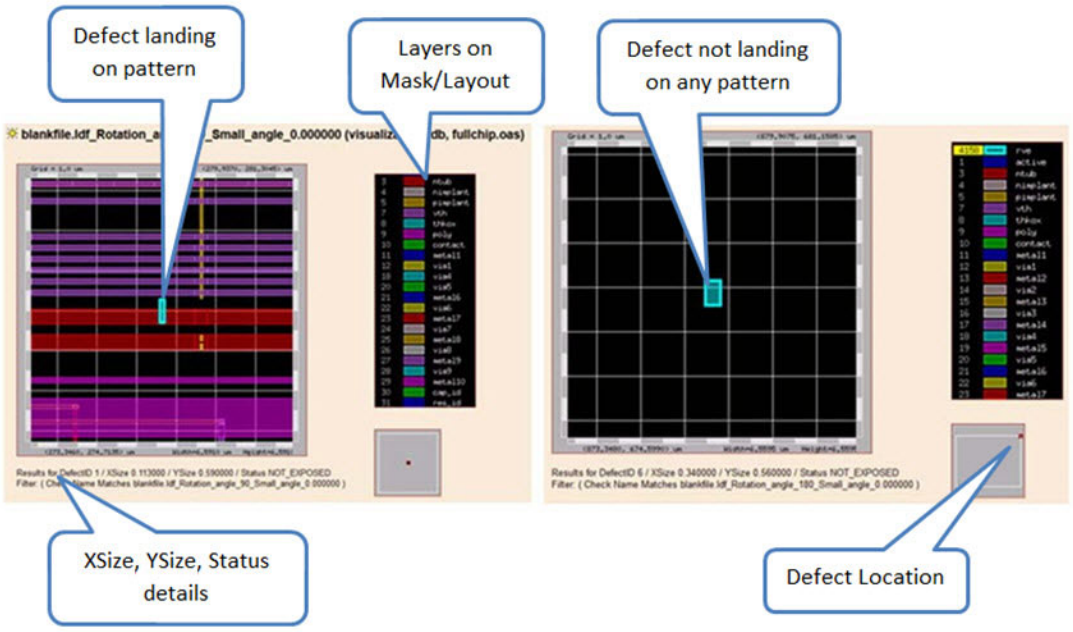


HTML Report

A spatial view of transformed defects can be visualized in an HTML report.

An example HTML report is shown in Figure 3-10.

Figure 3-10. HTML Report

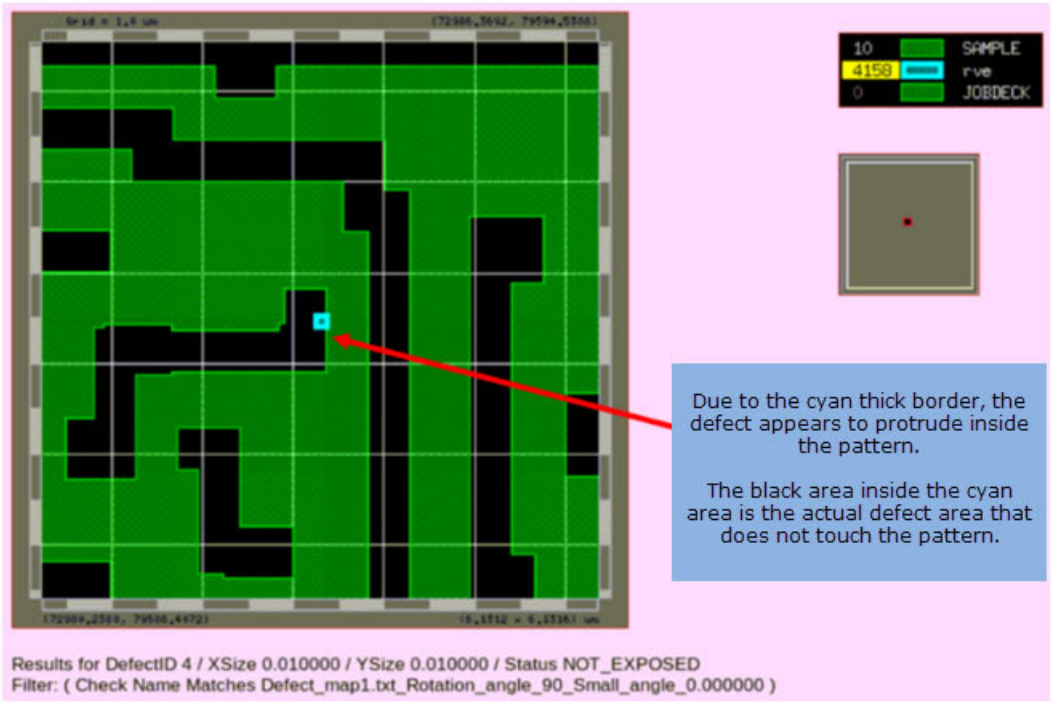


The window size for an image clip in the report can be defined using the HTML Report **Window Size** setting. Refer to "Specifying Defect Avoidance Run Settings" on page 31 for details.

In the case of OASIS and MEBES job decks, only data from specified layers (layers considered for defect avoidance) are captured in the clip image inside the report. Refer to "Specifying Inputs and Outputs" on page 25 for details on how to specify mask data layers.

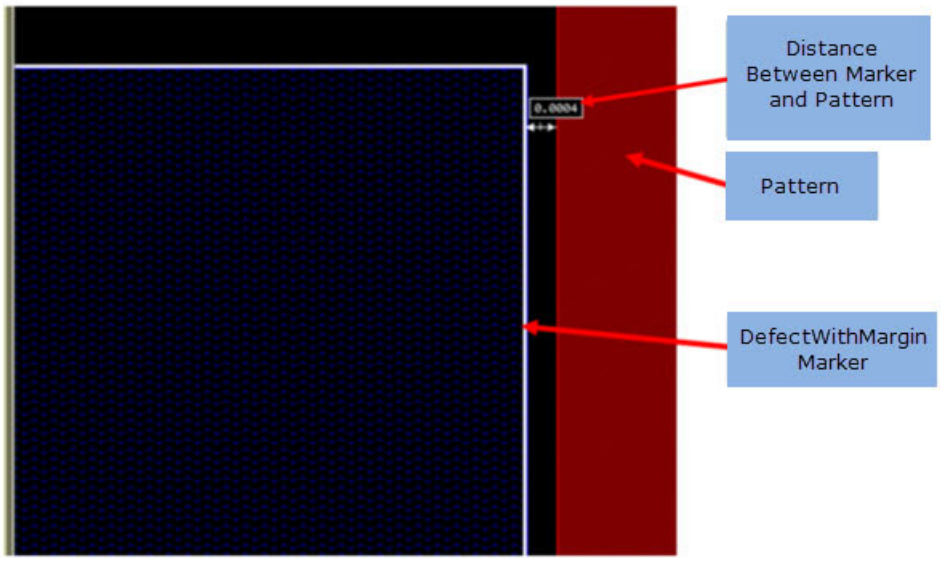
The marker placed in the layout is a DefectMarker marker. On occasion, due to a thick border of a marker, the marker can appear to be protruding inside the pattern (as shown in the following figure) but does not actually protrude.

Figure 3-11. DefectMarker in HTML Report



In these cases, use Calibre RVE to cross probe the marker in the layout as illustrated in the following figure:

Figure 3-12. Distance Between Pattern and DefectWithMargin Marker



The HTML Report can be generated independent of a Calibre MDPDefectAvoidance run using the calibre invocation command as described in “Generating an HTML Report With the Calibre Command” on page 75.

Generating an HTML Report With the Calibre Command

You can generate an HTML report independent of a Calibre MDPDefectAvoidance run using the Calibre invocation command.

Prerequisites

- A Calibre MDPDefectAvoidance run generates the following files inside an output report directory that are required for HTML report generation:
  - *db_map.txt* — Contains RDB and design file paths along with other information.
  - *html_report.ini* — Contains HTML report generation related parameters such as ZoomLayoutFactor and LayoutWidth.
  - *visualization.rdb.cto* — Generated only if the mask_data_layer parameter is specified in the parameter file. This file contains layers to be shown in the report.

Procedure

To generate the HTML report using the calibre command, use the following syntax:

```
$MGC_HOME/bin/calibre -rve -drc \
   -input <db-map-file> \
   -report <html_report_ini_file> \
   -output <html_report_directory>
```

For example, a Calibre MDPDefectAvoidance run has generated HTML input files inside of the directory *da_report_20200408_141704*. To generate the report:

```
$MGC_HOME/bin/calibre -rve -drc \
   -input da_report_20200408_141704/db_map.txt \
   -report da_report_20200408_141704/html_report.ini \
   -output htmlreportdir
```

A directory *htmlreportdir* is created and any report-related files are placed there. Refer to the *Calibre Interactive User's Manual* and the *Calibre RVE User's Manual* for details on DRC HTML reporting.

Note

For OASIS and extended job decks, HTML report generation can be sped up by using existing OASIS index and bin files. Set the following environment variables:

- setenv MDP_VBOASIS_INJECTION 1
- setenv MDP_VBOASIS_INJECTION_SIZE <*bin_file_size*> — Set the value of the bin file size to the bin size used when generating the bin files. The variable

CALIBRE_MDPDA_OASIS_BINSIZE is used to generate bin files of the desired size.

- setenv MDP_VBOASIS_INJECTION_PRESERVE 1

Generating an HTML Report in Parallel Using Remotes

Once a defect avoidance run is completed, you can generate HTML reports in parallel by providing an output directory created by the defect avoidance run.

Procedure

At a command prompt, use the following command:

```
$MGC_HOME/bin/calibremdpv -a da htmlreport <param_file> <report_dir>
-remotemachinesfile <remotemachinesfile> [ -configfile
<config_file>]
```

where:

- *param_file* — Specifies the parameter file use by for the defect avoidance run.
- *report_dir* — Specifies the report directory generated by the defect avoidance run.
- *remotemachinesfile* — Specifies the list of remote machines
- *config_file* — Specifies the configuration file used by the defect avoidance run.

Examples

In the following example, *params.txt* is the parameter file used for the defect avoidance run, *da_report_20200426_100333* is the output directory generated by the defect avoidance run, and *da_config.xml* is the run configuration file.

```
$MGC_HOME/bin/calibremdpv -a da htmlreport params.txt
da_report_20200426_100333 -remotemachinesfile
<remotemachinesfile>$MGC_HOME/bin/calibremdpv –a da htmlreport params.txt
da_report_20200426_100333 -remotemachinesfile <remotemachinesfile>
-configfile da_config.xml
```

The tool then checks the following directories for input files required for HTML report generation:

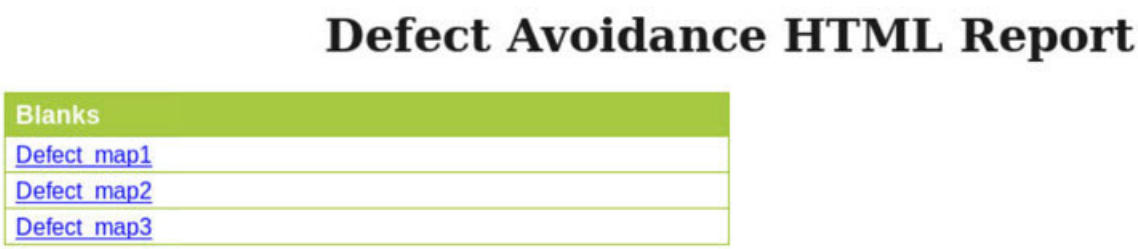
```
da_report_20200426_100333/da_report_Defect_map1/
da_report_20200426_100355/

da_report_20200426_100333/da_report_Defect_map2/
da_report_20200426_100457/

da_report_20200426_100333/da_report_Defect_map3/
da_report_20200426_100533/
```

A wrapper file */htmlreport/index.htm* is generated inside the output directory *da_report_20200426_10033*. The wrapper file *index.htm* contains links to each blank's HTML report as shown in following image.

Figure 3-13. HTML Report Main Page



The order of blanks in the wrapper HTML file might not in the same order as specified in the run configuration file.

You can also generate an HTML report in parallel during a defect avoidance run by setting the create_html_report parameter to true in the Calibre MDPDefectAvoidance parameter file.

OASIS Report

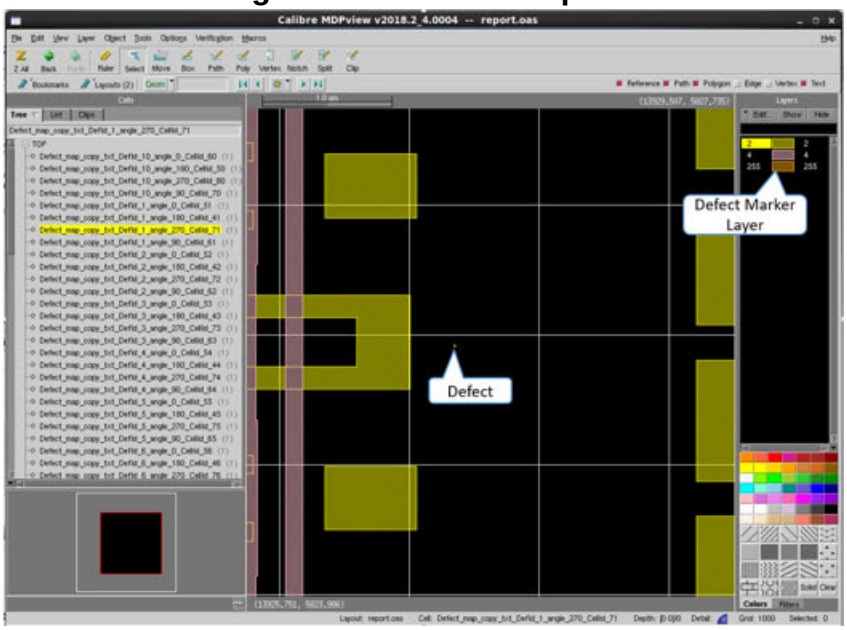
The output transformed defects from Calibre MDPDefectAvoidance can be visualized in an OASIS report.

A single OASIS file is generated per Calibre MDPDefectAvoidance run. The report contains similar information as the HTML report, however, where the HTML information is static, the OASIS report can be opened in a layout viewer such as Calibre MDPview and provides better flexibility for output review and results validation.

The size of each transformed defect is the defect size plus six microns, both horizontally and vertically. For example, if the defect width is 0.3 microns and the height is 0.2 microns, then the

defect clip size is 6.3 microns horizontally and 6.2 microns vertically. The following figure illustrates an example of the report:

Figure 3-14. OASIS Report



In this figure, the left side of the viewer window illustrates a cell number under the TOP cell. Each cell indicates one output defect location. The naming format of each cell is:

<DefectMapFile>_DefId_<defect-id>_angle_<rotation-angle>_CellId_<cell-id>

Layer information is displayed on the right side of the viewer window. A defect marker layer is added to represent the defect. The default defect marker layer is 255, which can be configured from the *$MGC_HOME/ pkgs/iccalibreda/pvt/calibredaconfig/da_config.xml* file. Refer to "Run Configuration File Format" on page 106 for details.

The shapes of different layers (in the cases of OASIS or MEBES JOBDECK input mask data) are separated in the OASIS report if the layer information is supplied through mask data layers. Refer to the mask_data_layer parameter "Parameters File Format" on page 87 for further information how layer information is supplied. If the mask data layers information is not supplied, then the shapes are merged into a single layer in the OASIS report.

Terminate HTML Report Generation

HTML report generation can be terminated using the following environment variable.

```
setenv CALIBRE_DA_HTML_REPORT_GENERATION_TIMEOUT <time_in_sec>
```

Set this variable to a positive non-zero integer value (in seconds) to set the timeout value. If report generation exceeds the time specified, the run is terminated. If report generation is aborted, any partial HTML report generated might be invalid. If this environment variable is not

set or the value supplied is invalid, report generation is not terminated. If you specify floating point values (such as 5.3 seconds or 5.8 seconds) for timeout, the value is truncated to the integer value in seconds (for example, 5.3 and 5.8 are truncated to 5 seconds). A message is generated in the log when a run terminates due to a timeout.

Log File

Calibre MDPDefectAvoidance activity output file

Calibre MDPDefectAvoidance generates a text file containing console outputs generated during its run.

A text log file named *cal_mdpda.log* is created in the current working directory per Calibre MDPview session. Console outputs generated during the session are appended to the log file. Once the session is closed and restarted, the older log file contents are overwritten with new session logs. Log files contain information that can help when debugging runtime issues or any other problems with the Calibre MDPDefectAvoidance run.

Though Calibre MDPDefectAvodiance generates the log file *cal_mdpda.log* in batch mode execution, it is recommended that you redirect the terminal log to a file at your desired location. The *cal_mdpda.log* files may not capture some messages output on the terminal window by Calibre MDPview during the start of a defect avoidance run. For example:

```
$MGC_HOME/bin/calibremdpv -remotefile remotes.crp -a da avoid params.txt
| tee test.log
```

In this case, the *test.log* file includes the log messages output on the terminal window.

Format

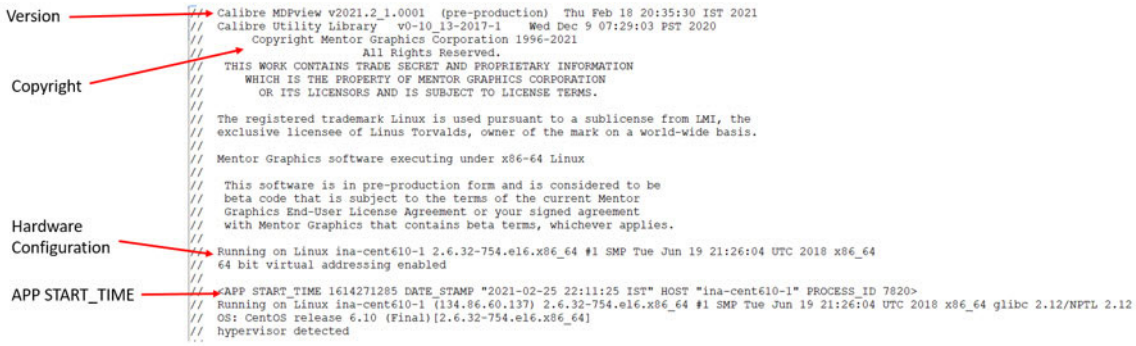
A text file called *cal_mdpda.log*.

Parameters

- Header Information

  Contains initial run information such as the product version, copyright, hardware configuration, and application start time.

Figure 3-15. Header Information



- License Information

  Contains the product licenses consumed to perform the defect avoidance run.

Figure 3-16. License Information

```
//  Running on 24 CPUs
//  Your timebomb license will expire: Thu May 13 2021

//  mentorall_s license acquired (calmdpview requested).
//  MDPview running on 24 cores
Waiting to check out MDPDefectAvoidance license(s) ...
//  mentorall_s license acquired (caleuvda requested).
//  mentorall_s license acquired (caleuvda requested).
//  mentorall_s license acquired (caleuvda requested).
//  mentorall_s license acquired (caleuvda requested).
//  mentorall_s license acquired (caleuvda requested).
//  mentorall_s license acquired (caleuvda requested).
//  mentorall_s license acquired (caleuvda requested).
//  MDPview running on 24 cores
//  MDPDefectAvoidance running on 24 cores
Acquired MDPDefectAvoidance license(s).
```

- Command Line

  Contains the command line executable used for the defect avoidance run.

Figure 3-17. Command Line

```
MDPDefectAvoidance log file: /ina/inascratch/rsoni/DefectAvoidance/CustomerIssues/Samsung/57.DA_reverse_tone/New_Jobdeck/cal_mdpda.log

Running: /home/icbuild/iwas/calibre_2021_2/ic/ic_superproj/aoi/Mgc_home//pkgs/icwb/pvt/calibrewb  -mdp -remotefile remotes_2021_2.crp -a da avoid params.txt
```

- Remote Host Configuration

  Contains the configuration options set for remote hosts for the defect avoidance run.

Figure 3-18. Remote Host Configurations

```
Defect Avoidance Environment variables:
CALIBRE_CMD_LINE=/home/icbuild/iwas/calibre_2021_2/ic/ic_superproj/aoi/Mgc_home//pkgs/icwb/pvt/calibrewb  -mdp -remotefile remotes_2021_2.crp -a da avoid params.txt
CWB_SUPRESS_BANNER=0
CWB_DISABLE_BANNER_SUPPRESSION=1
MGC_HOME=/home/icbuild/iwas/calibre_2021_2/ic/ic_superproj/aoi/Mgc_home/
HOME=/home/rsoni
CALIBRE_MDPDA_OASIS_BINSIZE=
CALIBRE_PLACEMENT_REPETITION_ENABLE=
MDP_VBOASIS_INJECTION=
MDP_VBOASIS_INJECTION_SIZE=
MDP_VBOASIS_INJECTION_PRESERVE=
CALIBRE_DA_HTML_REPORT_GENERATION_TIMEOUT=
CALIBRE_MTFLEX_INDEXING_ENABLE=18011984
CALIBRE_DISTRIBUTED_INDEXING=
CALIBRE_OASIS_INDEX_01=
CALIBREWB_MTFLEX_HYPERTHREADING=
CALIBRE_ENABLE_CBLOCK_DISTRIBUTED=
```

- Defect Avoidance Environment Variables

  Lists all the environment variables set for the defect avoidance run. For example, the following environment variables are set before defect avoidance execution along with other environment variables required for various operations:

  ```
  setenv CWB_SUPRESS_BANNER 0
  setenv CWB_DISABLE_BANNER_SUPPRESSION 1
  ```

  These variables are listed in the log file section shown in the following figure.

Figure 3-19. Defect Avoidance Environment Variables

```
Defect Avoidance Environment variables:
CALIBRE_CMD_LINE=/home/icbuild/iwas/calibre_2021_2/ic/ic_superproj/aoi/Mgc_home//pkgs/icwb/pvt/calibrewb  -mdp -remotefile remotes_2021_2.crp -a da avoid params.txt
CWB_SUPRESS_BANNER=0
CWB_DISABLE_BANNER_SUPPRESSION=1
MGC_HOME=/home/icbuild/iwas/calibre_2021_2/ic/ic_superproj/aoi/Mgc_home/
HOME=/home/rsoni
CALIBRE_MDPDA_OASIS_BINSIZE=
CALIBRE_PLACEMENT_REPETITION_ENABLE=
MDP_VBOASIS_INJECTION=
MDP_VBOASIS_INJECTION_SIZE=
MDP_VBOASIS_INJECTION_PRESERVE=
CALIBRE_DA_HTML_REPORT_GENERATION_TIMEOUT=
CALIBRE_MTFLEX_INDEXING_ENABLE=18011984
CALIBRE_DISTRIBUTED_INDEXING=
CALIBRE_OASIS_INDEX_01=
CALIBREWB_MTFLEX_HYPERTHREADING=
CALIBRE_ENABLE_CBLOCK_DISTRIBUTED=
```

- MTflex Remote Host Usage

  Contains usage information on each remote host used by MTflex for the defect avoidance run. The following environment variable must be set to print RSS information.

  ```
  setenv CALIBRE_ALWAYS_REPORT_RSS 1
  ```

  LVHEAP information is available in the primary host and remote logs of MTflex runs when the following parameters are set in the host configuration file for MTflex runs.

Figure 3-20. LVHEAP Settings

```
LAUNCH AUTOMATIC
LOCAL HOST DIR /ina/remote_logs
MONITOR LOCAL LOAD MEMORY INTERVAL 10
MONITOR REMOTE MEMORY INTERVAL 10
MONITOR UTILITY
LAUNCH WAIT 10
```

Figure 3-21. MTflex Remote Host Usage

```
MTflex remote host usage
 REMOTE CPU ina-rhel6-1: PID = 2699, ID = 2, CPU TIME = 9, STATUS = OK(0), LVHEAP = 2/9/13, RX = 8, TX = 0, RSS = 57
 REMOTE CPU ina-rhel6-1: PID = 2557, ID = 1, CPU TIME = 0, STATUS = OK(0), LVHEAP = 1/7/11, RX = 8, TX = 0, RSS = 55
 REMOTE CPU ina-rhel6-1: PID = 2701, ID = 4, CPU TIME = 8, STATUS = OK(0), LVHEAP = 2/9/13, RX = 8, TX = 0, RSS = 57
 REMOTE CPU ina-rhel6-1: PID = 2695, ID = 3, CPU TIME = 9, STATUS = OK(0), LVHEAP = 2/9/13, RX = 8, TX = 0, RSS = 57
 REMOTE CPU ina-rhel6-1: PID = 2698, ID = 5, CPU TIME = 6, STATUS = OK(0), LVHEAP = 2/9/13, RX = 8, TX = 0, RSS = 58
 REMOTE CPU ina-rhel6-1: PID = 2697, ID = 6, CPU TIME = 0, STATUS = OK(0), LVHEAP = 1/7/11, RX = 4, TX = 0, RSS = 54
 REMOTE CPU ina-rhel6-1: PID = 2700, ID = 7, CPU TIME = 0, STATUS = OK(0), LVHEAP = 1/7/11, RX = 4, TX = 0, RSS = 54
 REMOTE CPU ina-rhel6-1: PID = 2696, ID = 8, CPU TIME = 0, STATUS = OK(0), LVHEAP = 1/7/11, RX = 4, TX = 0, RSS = 54
 REMOTE CPU ina-rhel6-2: PID = 18851, ID = 9, CPU TIME = 0, STATUS = OK(0), LVHEAP = 1/7/10, RX = 7, TX = 0, RSS = 53
 REMOTE CPU ina-rhel6-2: PID = 18985, ID = 12, CPU TIME = 4, STATUS = OK(0), LVHEAP = 1/9/13, RX = 7, TX = 0, RSS = 56
 REMOTE CPU ina-rhel6-2: PID = 18984, ID = 13, CPU TIME = 9, STATUS = OK(0), LVHEAP = 2/9/13, RX = 7, TX = 0, RSS = 57
 REMOTE CPU ina-rhel6-2: PID = 18988, ID = 14, CPU TIME = 8, STATUS = OK(0), LVHEAP = 2/9/13, RX = 7, TX = 0, RSS = 57
 REMOTE CPU ina-rhel6-2: PID = 18983, ID = 16, CPU TIME = 3, STATUS = OK(0), LVHEAP = 1/9/13, RX = 5, TX = 0, RSS = 57
 REMOTE CPU ina-rhel6-2: PID = 18986, ID = 15, CPU TIME = 7, STATUS = OK(0), LVHEAP = 2/9/13, RX = 7, TX = 0, RSS = 57
 REMOTE CPU ina-rhel6-2: PID = 18987, ID = 17, CPU TIME = 0, STATUS = OK(0), LVHEAP = 1/7/11, RX = 3, TX = 0, RSS = 54
 REMOTE CPU ina-rhel6-2: PID = 18989, ID = 18, CPU TIME = 0, STATUS = OK(0), LVHEAP = 1/7/11, RX = 3, TX = 0, RSS = 54
 REMOTE CPU ina-rhel6-3: PID = 16122, ID = 10, CPU TIME = 0, STATUS = OK(0), LVHEAP = 1/7/10, RX = 7, TX = 0, RSS = 53
 REMOTE CPU ina-rhel6-3: PID = 16255, ID = 22, CPU TIME = 3, STATUS = OK(0), LVHEAP = 1/9/13, RX = 5, TX = 0, RSS = 57
 REMOTE CPU ina-rhel6-3: PID = 16259, ID = 19, CPU TIME = 9, STATUS = OK(0), LVHEAP = 2/9/13, RX = 7, TX = 0, RSS = 56
 REMOTE CPU ina-rhel6-3: PID = 16256, ID = 11, CPU TIME = 4, STATUS = OK(0), LVHEAP = 1/9/13, RX = 7, TX = 0, RSS = 56
 REMOTE CPU ina-rhel6-3: PID = 16260, ID = 21, CPU TIME = 6, STATUS = OK(0), LVHEAP = 2/9/13, RX = 7, TX = 0, RSS = 57
 REMOTE CPU ina-rhel6-3: PID = 16254, ID = 20, CPU TIME = 8, STATUS = OK(0), LVHEAP = 2/9/13, RX = 7, TX = 0, RSS = 57
 REMOTE CPU ina-rhel6-3: PID = 16257, ID = 23, CPU TIME = 0, STATUS = OK(0), LVHEAP = 1/7/11, RX = 3, TX = 0, RSS = 54
 REMOTE CPU ina-rhel6-3: PID = 16258, ID = 24, CPU TIME = 0, STATUS = OK(0), LVHEAP = 1/7/11, RX = 3, TX = 0, RSS = 54
```

  For details on the logs generated by MTflex, refer to the *Calibre Administrator's Guide*.

- Remote Host Summary

  Summarizes the total usage of remote hosts used for the defect avoidance run.

Figure 3-22. Remote Host Summary

```
Remote Host Summary
 REMOTE LVHEAP ina-rhel6-1: RCS = 96 TOTAL = 96
 REMOTE LVHEAP ina-rhel6-2: RCS = 97 TOTAL = 97
 REMOTE LVHEAP ina-rhel6-3: RCS = 97 TOTAL = 97
```

- APP_END_TIME

  Specifies the time the application run ends.

Figure 3-23. APP_END_TIME

```
//  <APP END_TIME 1614271311 DATE_STAMP "2021-02-25 22:11:51 IST" HOST "ina-cent610-1" PROCESS_ID 7820 ELAPSED 26>
```

Defect Avoidance Timers

A defect avoidance run logs CPU and REAL time taken by any operation. All the times reported are in seconds unless a different unit is explicitly specified. The printing of these timers is

controlled by the LogLevel node setting in the Calibre MDPDefectAvoidance run configuration file.

- PARAM_FILE_PARSE_TIME

  Specifies the time taken to parse the input parameter file.

Figure 3-24. PARAM_FILE_PARSE_TIME

```
PARAM_FILE_PARSE_TIME(seconds): CPU TIME = 0.0000  REAL TIME = 0.0025
```

- CONFIG_FILE_PARSE_TIME

  Specifies the time taken to parse the run configuration file.

Figure 3-25. CONFIG_FILE_PARSE_TIME

```
CONFIG_FILE_PARSE_TIME(seconds): CPU TIME = 0.0000  REAL TIME = 0.1118
```

- BLANK_FILE_PARSE_TIME

  Specifies the time taken to parse an input blank inspection report file.

Figure 3-26. BLANK_FILE_PARSE_TIME

```
BLANK_FILES_PARSE_TIME(seconds): CPU TIME = 0.0000  REAL TIME = 0.0091
```

- DESIGN_LOAD_TIME

  Specifies the time taken to load a design file such as an OASIS file or a job deck in the primary host machine.

Figure 3-27. DESIGN_LOAD_TIME

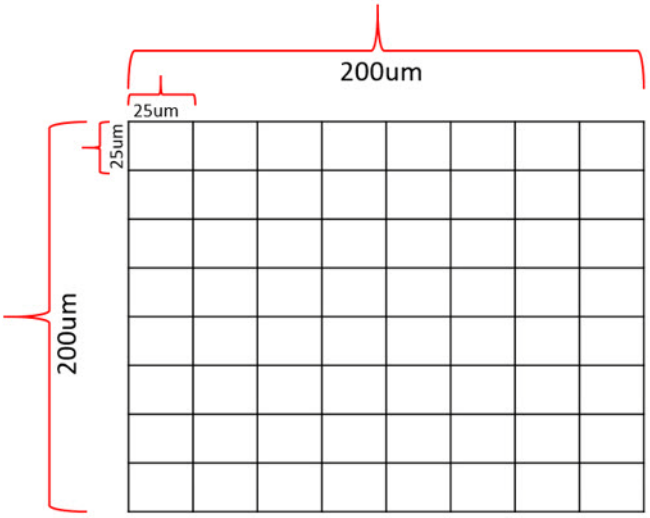
```
DESIGN_LOAD_TIME(seconds): CPU TIME = 0.0100  REAL TIME = 0.0645
```

- SECTION_PROCESSING_TIME

  In an MTflex run, the total query window (decided by shift x and y, defect size, and setbacks) is divided into smaller sections and each remote host processes only a small amount of data from all the defects. The number of sections is determined by the section's x and y parameters specified in the defect avoidance parameter file.

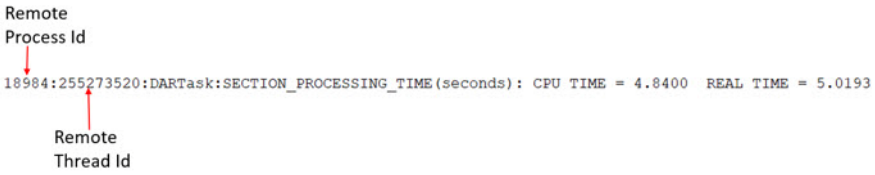
  For example, the window to process is specified as 200x200 um and the section's x and y is set to 8x8. In that case, each remote machine processes only a 25x25 um region at a time from all defects. If the MTflex run is started with 64 cores, then all 25x25um windows are processed in parallel.

Figure 3-28. Defect Avoidance Section Processing



The SECTION_PROCESSING_TIME is the cumulative time taken by each remote to process a section. This time includes the time to query and process queried data for one section such a 25x25 um region.

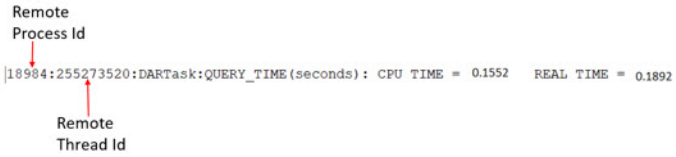
Figure 3-29. SECTION_PROCESSING_TIME



- QUERY_TIME

  Specifies the time taken by the remote machine to query one section's window data from the design for only one defect.

Figure 3-30. QUERY_TIME



- TOTAL_MDPDEFECTAVOIDANCE_TIME

  Specifies the cumulative time taken by a defect avoidance run, the exposed finder, HTML report generation, and OASIS report generation.

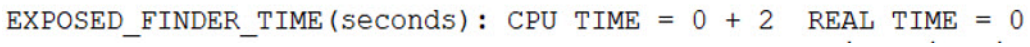
Figure 3-31. TOTAL_MDPDEFECTAVOIDANCE_TIME

```
TOTAL_MDPDEFECTAVOIDANCE_TIME(seconds): CPU TIME = 0 + 100  REAL TIME = 11
```

- EXPOSED_FINDER_TIME

  Specifies the time taken to cross verify if a defect is overlapping with any pattern after applying the shifts resulting from a defect avoidance run.

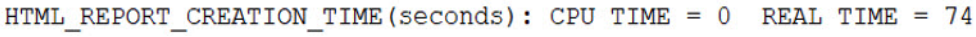
Figure 3-32. EXPOSED_FINDER_TIME



- HTML_REPORT_CREATION_TIME

  Specifies the time taken to generate the HTML report. The report is generated inside the *htmlreport* directory in the output directory.

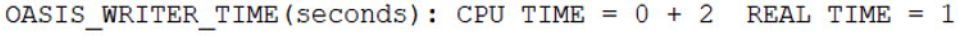
Figure 3-33. HTML_REPORT_CREATION_TIME



- OASIS_WRITER_TIME

  Specifies the time taken to generate an OASIS report. The report *report.oas* is located inside the output directory.

Figure 3-34. OASIS_WRITER_TIME



Calibre MDPDefectAvoidance Files

The parameters file and run configuration file directly affect Calibre MDPDefectAvoidance.

- **Parameters File** — All the inputs, outputs and settings-related values are passed to Calibre MDPDefectAvoidance using a parameter file. This is generated when you save from the Calibre MDPDefectAvoidance tool.
- **Run Configuration File** — Certain default behaviors and settings with Calibre MDPDefectAvoidance are controlled in an XML configuration file, *da_config* (included with your Calibre distribution). This includes debugging information, reticle radius, database unit information, and consideration of exposed defects, maximum solution width, and maximum exposed defects count to compute the usability factor for a blank-design pair.
- **Pairing Optimizer Parameters File** — The contents that are displayed in the Pairing Optimizer tool can be modified using an XML parameter file.

Parameters File Format

Calibre MDPDefectAvoidance run settings file

The parameters file contains information regarding the input mask or layout data file, blank mask inspection (defect data) files, and other run control settings that are loaded into Calibre MDPDefectAvoidance for a run.

This file can be generated and modified through the Calibre MDPDefectAvoidance tool in Calibre MDPview (see “Saving and Loading a Parameter File” on page 40).

Format

There are two different formats that Calibre MDPDefectAvoidance will accept input from:

- A text (.txt) parameter file. The following illustrates an example of a text parameter file.

```
daparamfile_version 2
max_shift_x 50
max_shift_y 50
resist_type PCAR
defect_placement_type OPAQUE
small_angle_step_count 0
small_angle_rotation_range 0
clear_edge_setback 0.0125
opaque_edge_setback 0.0125
global_defect_size_threshold 0.01
global_defect_area_threshold 0.0001
global_defect_type_filters 1, 2, 1A, 2A
mask_data_name MASK1
mask_data_file "/home/user/demo.jb"
mask_data_type MEBESJOB
mask_data_layer 1:0 2:0
mask_data_layer_rule 1:0 100 -defectsizetolerance 0.002
mask_data_layer_rule 2:0 50 -defectsizetolerance 0.0005
blank_defect_map_source_directory "/home/user/basedir"
mask_blank_defect_data "/home/user/demoklarf.klarf" klarf
mask_blank_defect_data "/home/user/demoldf.ldf" ldf
mask_blank_defect_data "/home/user/demotxt.txt" txt
DBfile da_demo.db
overwrite true
create_html_report false
create_oasis_report true
report_directory "/home/user/reports"
defect_type 1 priority HIGH
defect_type 2 priority MEDIUM
defect_type 3 priority LOW
do_care_region_file "/home/user/docare.txt"
do_not_care_region_file "/home/user/donotcare.txt"
```

- An XML (.xml) parameter file. Figure 3-35 illustrates an example of an XML parameter file.

Figure 3-35. XML Parameter File (Part One)

```
<?xml version='1.0' encoding='utf-8'?>
<DAParamFile version="3">
 <InputFiles>
  <BlankDefectMapSourceDirectory>/home/user/basedir</BlankDefectMapSourceDirectory>
  <MaskBlankInfoList>
   <MaskBlankDefectDataInfo>
    <MaskBlankFile>/home/user/demoklarf.klarf</MaskBlankFile>
    <MaskBlankType>klarf</MaskBlankType>
   </MaskBlankDefectDataInfo>
   <MaskBlankDefectDataInfo>
    <MaskBlankFile>/home/user/demoldf.ldf</MaskBlankFile>
    <MaskBlankType>ldf</MaskBlankType>
   </MaskBlankDefectDataInfo>
   <MaskBlankDefectDataInfo>
    <MaskBlankFile>/home/user/demotxt.txt</MaskBlankFile>
    <MaskBlankType>txt</MaskBlankType>
   </MaskBlankDefectDataInfo>
  </MaskBlankInfoList>
  <MaskDataInfo>
   <MaskDataFile>/home/user/demo.jb</MaskDataFile>
   <MaskDataName>MASK1</MaskDataName>
   <MaskDataType>MEBESJOB</MaskDataType>
   <MaskDataLayer>1:0, 2:0</MaskDataLayer>
   <MaskDataLayerRules>
    <MaskDataLayerRule>
     <Layer>1:0</Layer>
     <Priority>100</Priority>
     <DefectSizeTolerance>0.002</DefectSizeTolerance>
    </MaskDataLayerRule>
    <MaskDataLayerRule>
     <Layer>2:0</Layer>
     <Priority>50</Priority>
     <DefectSizeTolerance>0.0005</DefectSizeTolerance>
    </MaskDataLayerRule>
   </MaskDataLayerRules>
  </MaskDataInfo>
  <DoCareRegionFile>/home/user/docare.txt</DoCareRegionFile>
  <DoNotCareRegionFile>/home/user/donotcare.txt</DoNotCareRegionFile>
 </InputFiles>
```

Figure 3-36. XML Parameter File (Part Two)

```
<Settings>
 <PlacementParameters>
  <MaxShiftX>50</MaxShiftX>
  <MaxShiftY>50</MaxShiftY>
  <SmallAngleRotationRange>10</SmallAngleRotationRange>
  <SmallAngleStepCount>1</SmallAngleStepCount>
  <DefectPlacementType>OPAQUE</DefectPlacementType>
  <ResistType>PCAR</ResistType>
  <ClearEdgeSetback>0.05</ClearEdgeSetback>
  <OpaqueEdgeSetback>0.05</OpaqueEdgeSetback>
  <XSections>2</XSections>
  <YSections>2</YSections>
 </PlacementParameters>
 <DefectPriorities>
  <Low>3</Low>
  <Medium>2</Medium>
  <High>1</High>
 </DefectPriorities>
 <DefectFilters>
  <GlobalDefectSizeThreshold>0.01</GlobalDefectSizeThreshold>
  <GlobalDefectAreaThreshold>0.0001</GlobalDefectAreaThreshold>
  <GlobalDefectTypeFilters>1, 2, 1A, 2A</GlobalDefectTypeFilters>
 </DefectFilters>
 <OutputInformation>
  <DatabaseFile>da_demo.db</DatabaseFile>
  <OverwriteDatabase>true</OverwriteDatabase>
  <CreateHtmlReport>false</CreateHtmlReport>
  <CreateOasisReport>true</CreateOasisReport>
  <ReportDirectory>/home/user/reports</ReportDirectory>
 </OutputInformation>
 </Settings>
</DAParamFile>
```

Parameters

Input Parameters

These parameters specify the inputs for a Calibre MDPDefectAvoidance run, including specifying mask data and blank mask inspection file information, as well as identifying critical and non-critical layout regions for analysis.

- daparamfile_version *num*

  An optional parameter file version. The version number is used for backward compatibility. All previous version of parameter files should continue to work with the current version of Calibre MDPDefectAvoidance. The current parameter file version is 4. The default setting is 1 if not otherwise specified.

  Equivalent XML parameter name: DAParamFile

- **mask_data_name *mask_identifier***

  A required unique identifier for the input mask. The mask identifier can be any string that uniquely identifies the mask.

  Equivalent XML parameter name: MaskDataName

- **mask_data_file *input_mask_path***

  A required parameter that specifies the path of the input mask file.

Equivalent XML parameter name: MaskDataFile

- **mask_data_type *input_mask_type***

  A required parameter that specifies the type of the input mask file. The following types are supported: OASIS, VSB, VSBJOB, MEBES, MEBESJOB, and JEOL.

  Equivalent XML parameter name: MaskDataType

- **layout_name *name***

  This parameter is required when **mask_data_type** is VSBJOB. The layout name should correspond to the MDS (Mask Drawing Script) of the selected VSBJOB file.

  Equivalent XML parameter name: LayoutNameForVSBJOB

- do_care_region_file *docare_file*

  An optional parameter that specifies the do-care regions, or critical regions that should be examined by the run. Regions are specified in the following sequence: bottom-left X coordinate, bottom-left Y coordinate, top-right X coordinate and top-right Y coordinate. Coordinates should be specified in microns. If only do-care regions are specified, then the other regions are considered as do-not-care regions.

  You can specify the critical layout regions using do-care region text files. Defects appearing on these regions may affect the features and should be avoided.

  Equivalent XML parameter name: DoCareRegionFile

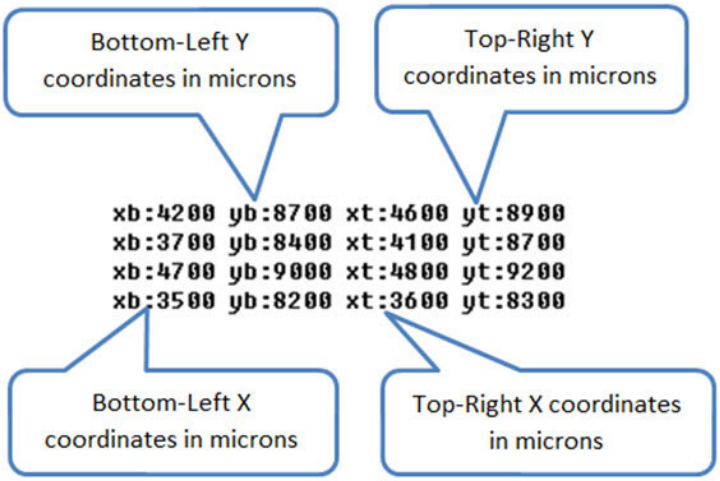
- do_not_care_region_file *do_not_care_file*

  This optional parameter specifies the do-not-care regions, or non-critical regions that should be ignored during the run. Regions are specified in the following sequence: bottom-left X coordinate, bottom-left Y coordinate, top-right X coordinate, and top-right Y coordinate. Coordinates should be specified in microns.

  You can specify the non-critical layout regions using do-not-care region text files. Defects appearing on these regions are ignored as they do not impact the yield. If only do-not-care regions are specified, then the other regions are considered as do-care regions. An example text file is shown in Figure 3-37.

  Equivalent XML parameter name: DoNotCareRegionFile

Figure 3-37. Do-Care and Do-Not-Care Regions Example Format



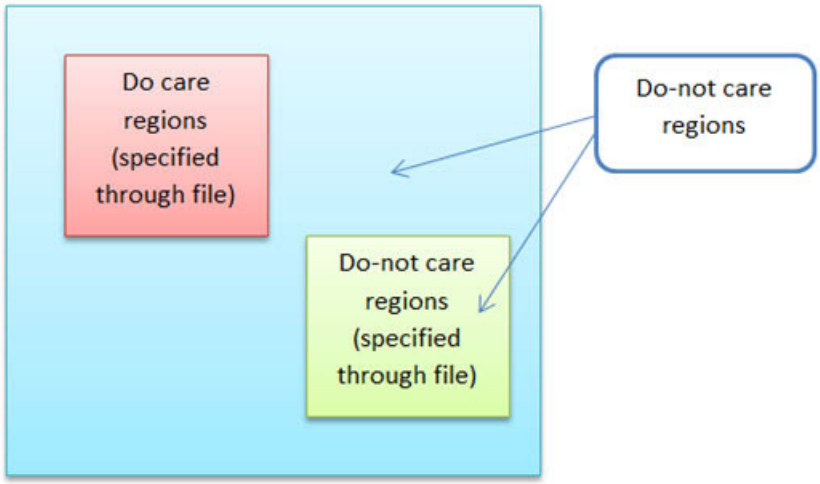
The following apply:

- If both do-care and do-not-care region files are not specified then the complete area is considered as a do-care region.
- If both do and do-not-care regions files are specified then do-care and do-not-care regions for defect avoidance are computed.

For example:

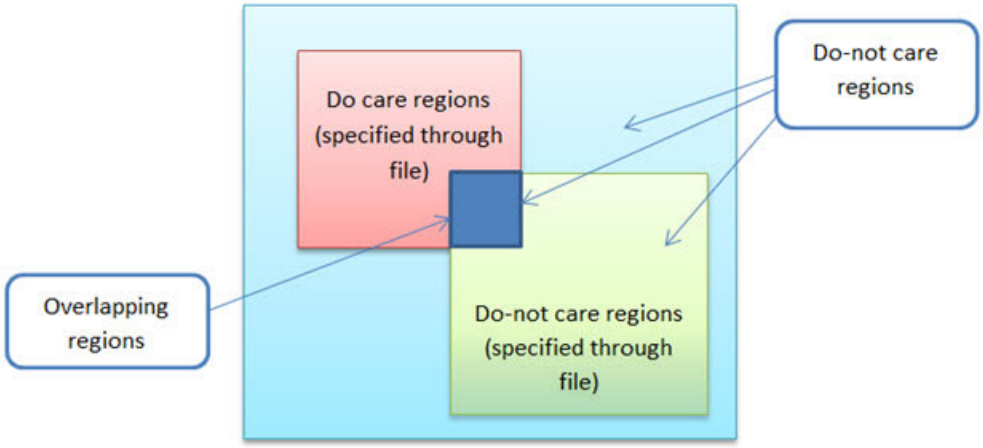
- If do-care and do-not-care regions do not overlap, then regions other than do-care regions are considered as do-not-care regions even they are not specified in the do-not-care regions file as shown in Figure 3-38.

Figure 3-38. Do and Do-Not-Care Regions Do Not Overlap



- If do and do-not-care regions overlap, then regions other than do-care regions and overlapping regions are considered as do-not-care regions as shown in Figure 3-39.

Figure 3-39. Do and Do-Not-Care Regions Overlap



- do_care_jdchips_file *do_care_chips_file*

  An optional parameter that specifies the do-care chip names or critical chip regions that should be examined by the run. If only a do-care chip name file is specified, then the other chips are considered as do-not-care chips.

  You can specify the critical layout regions using do-care chip names text files. Defects appearing on these regions can affect the features and should be avoided.

  Equivalent XML parameter name: DoCareChipsFile

- do_not_care_jdchips_file *do_not_care_chips_file*

  An optional parameter that specifies the do-not-care chip names. If only a do-not-care chip name file is specified, then the other chips are considered as do-care chips.

  You can specify the non-critical layout regions using do-not-care chips. Defects appearing on these regions are ignored as they do not impact the yield. If only do-not-care chips are specified, then the other chips are considered as do-care chips.

  Equivalent XML parameter name: DoNotCareChipsFile

  The following applies to do_care_jdchips_file and do_not_care_jdchips_file:

  - In chips text files, each chip name is specified on a separate line.
  - Chip names should be in the format of *XXXXXXXXX.XX* (9.2), which is the actual chip file name.
  - You can specify do-care and do-not-care regions in both the formats. Existing text files include Bounding Boxes and files containing chip names.
  - Chip name files are supported only for MEBES JOBDECK.
  - Chip names provided inside chip name files are used to extract chip bounding boxes, then bounding boxes are considered as do-care and do-not-care regions. The

following operations are performed to compute actual do-care and do-not-care regions in the case of an overlap between their bounding boxes:

- Do_care region: Do-care bounding box extracted from do-care chips
- Do-not-care region: (Do-not-care bounding box extracted from do-not-care chips) – (Do care bounding box extracted from do-care chips).

  - If do-care chips are specified, only do-care regions are calculated using do-care chip bounding boxes. Other chips are ignored and they are considered non-critical regions.

- **mask_blank_defect_data** ***blank_file_pathblank_type***

  A required parameter that specifies the details of the blank mask file including pathname (***blank_file_path***) and type (***blank_type***). The base name of ***blank_file_path*** is used as a unique identifier for the blank. For example, if ***blank_file_path*** is */home/user1/blank1.ldf* then *blank1.ldf* is used as the blank file identifier. At least one valid blank file is required to run Calibre MDPDefectAvoidance.

  Equivalent XML parameter names: MaskBlankFile, MaskBlankType.

- blank_defect_map_source_directory *base_dir_path_for_blanks*

  An optional parameter that provides the base directory for all blanks. When selecting a blank file from the GUI, this base directory is opened by default.

  Equivalent XML parameter name: BlankDefectMapSourceDirectory.

Control Parameters

These parameters specify defect avoidance controls such as allowable shifts, image tone, defect priorities and edge setbacks.

- **max_shift_x** ***val***

  A required parameter (an unsigned double) that specifies the furthest distance (in microns) the pattern can be shifted relative to the blank in X-dimension (left and right). The default value is 100 microns and the range is [0, 500].

  An error is issued if the value specified is more than 500 microns. A warning is displayed if the value is more than 200 microns but less than or equal to 500 microns.

  Equivalent XML parameter name: MaxShiftX.

- **max_shift_y** ***val***

  A required parameter (an unsigned double) that specifies the furthest distance (in microns) the pattern can be shifted relative to the blank in Y-dimension (top and bottom). The default value is 100 microns and the range is [0, 500].

  An error is issued if the value specified is more than 500 microns. A warning is displayed if the value is more than 200 microns but less than or equal to 500 microns.

  Equivalent XML parameter name: MaxShiftY.

- major_rotations *val*

  An optional parameter specifying major rotational angles with the five allowed values: All, 0, 90, 180 and 270. Major rotations are used to find the best defect avoidance solutions by rotation blank masks with different angles. A combination of major rotations such as “0, 180” can be used to find a solution using only 0 and 180 angle rotations. By default, all major angles 0, 90, 180 and 270 are used.

  Equivalent XML parameter name: MajorRotations.

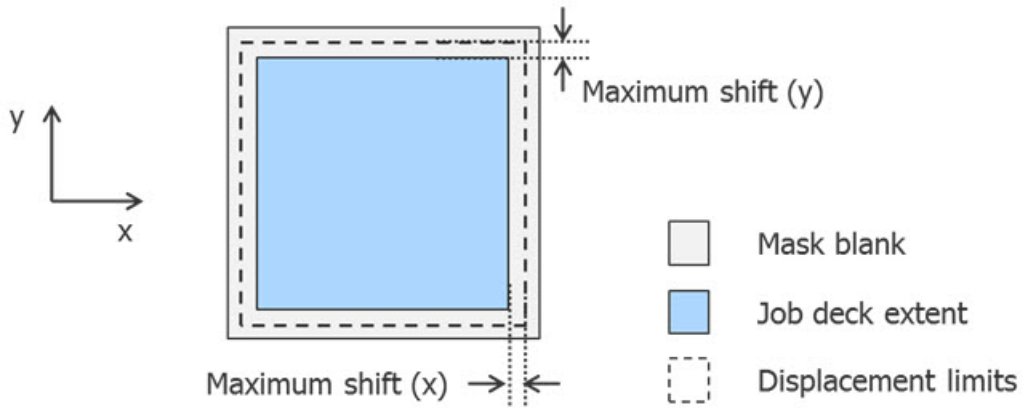
- small_angle_rotation_range *val*

  An optional parameter (an unsigned double) that specifies the length (in microns) of the arc subtended by two radii after the micro rotation. The default value is 0.

  Using small_angle_rotation_range, Calibre MDPDefectAvoidance computes the maximum allowable rotation. The allowable small-angle-rotation values are determined by ensuring that the corners of the pattern do not move outside the allowed displacement limits (as illustrated by the dotted line in Figure 3-40).

  Equivalent XML parameter name: SmallAngleRotationRange.

Figure 3-40. Definition of Maximum Shift in X and Y Directions



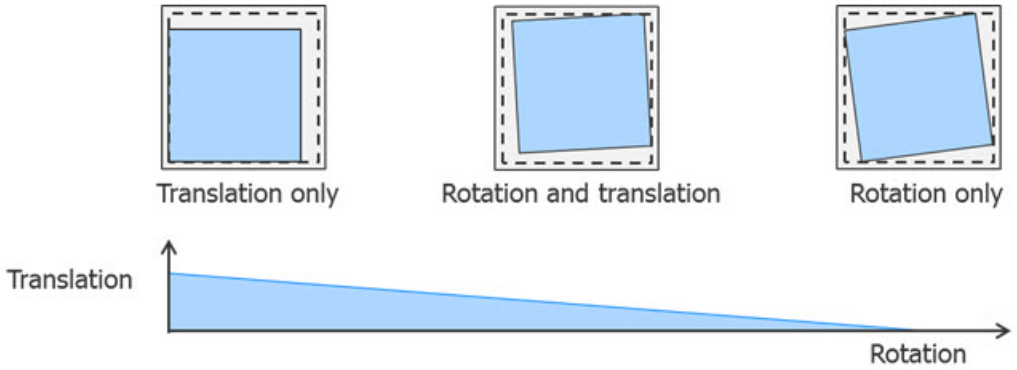
- small_angle_step_count *val*

  An optional parameter (an unsigned integer) that specifies the number of steps to be used for micro rotation. The default value is 0.

  The small angle rotation is controlled by the number of steps that should be used. If you specify 3 steps, the tool calculates the angle of rotation (in this case, 0.05, 0.1 and 0.15 degrees) and rotates the pattern both clockwise and counter-clockwise by each of these angles.

  In addition, since rotating using a small angle and then shifting the pattern by the maximum allowable translation moves the corner of the pattern outside of the allowed region, the tool automatically reduces the allowable translation for each rotation. This ensures that the pattern always stays within the displacement limits (as illustrated in Figure 3-41).

Figure 3-41. Relationship Between Rotation, Translation, and Displacement Limits



Equivalent XML parameter name: SmallAngleStepCount.

- **resist_type** {**PCAR** | **NCAR**}

  A required parameter that specifies the image tone type.

  PCAR — Drawn shapes are clear (reflective) on the mask. This is the default value.

  NCAR — Drawn shapes are opaque (absorber) on the mask.

  Equivalent XML parameter name: ResistType.

- **defect_placement_type** {**OPAQUE** |CLEAR | EITHER}

  A required parameter that specifies the region where defects should be placed such that it does not affect the patterns.

  OPAQUE — Defects are be placed in opaque (absorber) regions. This is the default value.

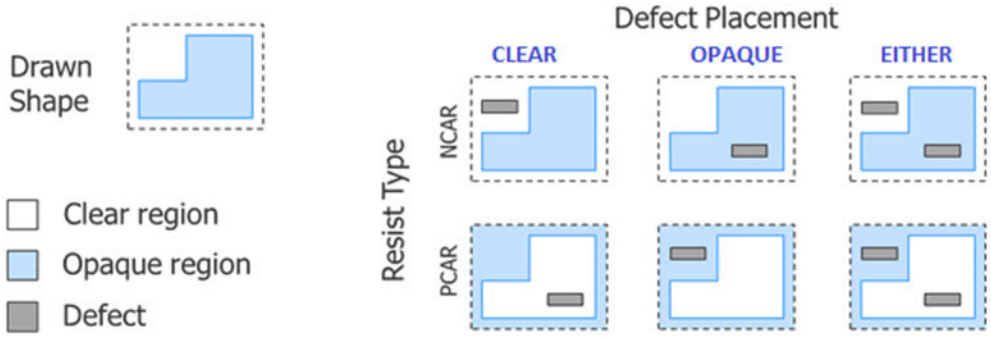
  CLEAR — Defects are be placed in clear (reflective) regions.

  EITHER — Defects are allowed to be placed in both opaque and clear regions.

  Used with the resist_type settings, this creates the six defect placement possibilities (valid defect avoidance locations) as illustrated in Figure 3-42.

  Equivalent XML parameter name: DefectPlacementType.

Figure 3-42. Resist Type and Defect Placement



- clear_edge_setback *val* opaque_edge_setback *val*

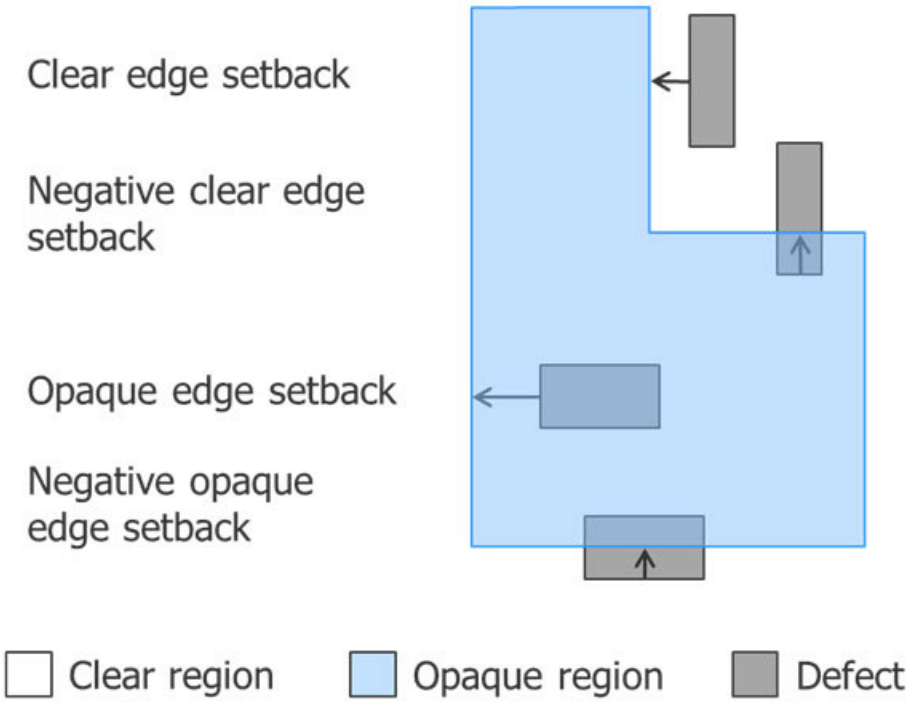
  Optional parameters (signed doubles) that specify the distance (in microns) from the edge of the drawn shape to the edge of the defect (as illustrated in Figure 3-43). Which parameter applies depends on where the defect is placed (as specified by the defect_placement_type parameter).

  Defects placed in clear regions use clear_edge_setback (the setback from the edge of the clear region where they are located).

  Defects placed in opaque regions use opaque_edge_setback (the setback from the edge of the opaque regions where they are located).

  Negative edge setbacks allow defects to protrude out of their region up to the amount of negative setback you specify.

Figure 3-43. Clear and Opaque Edge Setbacks



  For clear_edge_setback and opaque_edge_setback, the default value is 0.05 microns and the range is [-5.0, 5.0].

  Equivalent XML parameter names: ClearEdgeSetback, OpaqueEdgeSetback.

- defect_type *defect_class* priority {HIGH | MEDIUM | LOW}

  Specifies the priority of the defect class. The classification of the defect is extracted from the input blank mask file. There may be multiple entries of defect_type in the parameter file. Each entry specifies the priority of a single defect-class.

  LOW priority defects are less critical defects compared to MEDIUM priority defects that are, in turn, less critical than HIGH priority defects.

For example:

- LDF
  - defect_type 1 priority HIGH
  - defect_type 2 priority HIGH
  - defect_type 3 priority MEDIUM
  - defect_type 4 priority LOW
- KLARF
  - defect_type 1A priority HIGH
  - defect_type 2B priority HIGH
  - defect_type 3D priority MEDIUM
  - defect_type 4A priority LOW

Equivalent XML parameter name: DefectPriorities.

Layer Priorities Parameters

The layer priorities capability of Calibre MDPDefectAvoidance allows you to prioritize the defects based on the layer geometries they appear on by splitting input layers out into different layers or levels based upon their functional intent. These parameters are used only with MEBES job deck and OASIS design inputs.

- mask_data_layer [*layer*[:*datatype*]]...

  An optional parameter that specifies the design layers to be considered and loaded into Calibre MDPDefectAvoidance. All other layers present in the input design file are ignored.

  Equivalent XML parameter name: MaskDataLayer.

  For example:

  - MEBESJOB

    ```
    # Load layers 1, 2, 20, and 30 from a MEBES job deck.

    mask_data_layer  1 2 20 30
    ```

  - OASIS

    ```
    # Load layers 1 with data type 0, 2 with data type
    # 1, 20 with data type 1 and 30 with data type 0 from OASIS.

    mask_data_layer 1:0 2:1 20:1 30:0
    ```

- mask_data_layer_rule *layer_num data_type priority* -defectsizetolerance *size*

  An optional parameter that specifies a rule for a particular layer. If mask_data_layer_rule is not specified for any loaded layer, the default priority of 100 (most critical layer) and defect size tolerance of 0 um are used for layer processing.

  layer_num — Specifies the layer number.

  data_type — Specifies the data type of the layer. This applies only to the OASIS format.

  priority — Specifies the priority. The higher the number, the higher the priority (or critical) the layer. The maximum priority is 100. 100 is also the default priority for any layer.

  -defectsizetolerance *size* — If the defect size is more than 0.025 um, then the defect is considered for avoidance or else it is ignored. The size of the defect is extracted from the blank defect inspection report.

  For example:

  - MEBESJOB

```
# Layer 1 has priority 100 and defects of size less than 0.025 um
# are to be ignored.

mask_data_layer_rule 1 100 -defectsizetolerance 0.025

# Layer 20 has priority 99 and tolerance 0 um. All other layers
# loaded have a default priority 100 and tolerance 0 um.

mask_data_layer_rule 20 99
```

  - OASIS

```
# Layer 1 with data type 0 has priority 100 and defects of size
# less that 0.025 um are ignored.

mask_data_layer_rule 1:0 100 -defectsizetolerance 0.025

# Layer 20 with data type 1 has priority 99 and tolerance 0 um.
# All other layers loaded will have a default priority 100 and
# tolerance 0 um.
mask_data_layer_rule 20:1 99
```

  Equivalent XML parameter name: MaskDataLayerRules.

Order of Priority for Defects and Layers

Defect classification (defect_type parameter) and layers priority (mask_data_layer_rule) together provides the order in which defects can be processed to find the solution space. This is described in detail in “Order of Priority for Defects and Layers” on page 102

Defect Filters

These parameters control how defects are filtered in Calibre MDPDefectAvoidance.

- global_defect_size_threshold *val*

  An optional parameter (signed double) that specifies the defect size threshold (in microns) to filter the non-critical defects based on their size. The default value is 0. The size information is extracted from the mask inspection reports. None of these defects are considered for defect avoidance.

  Equivalent XML parameter name: GlobalDefectSizeThreshold.

- global_defect_area_threshold *val*

  An optional parameter (signed double) that specifies the defect area threshold (in microns) to filter the non-critical defects based on their area. The default value is 0. The area of defect is computed from defect sizes information extracted from the mask inspection reports. None of these defects are considered for defect avoidance.

  Equivalent XML parameter name: GlobalDefectAreaThreshold.

- global_defect_type_filters *defect_types*

  An optional parameter to filter the non-critical defects based on their types. Defect types from various mask inspection formats can be specified with global_defect_type_filters. The defect types should be separated by a comma (**,**). The default value is an empty string. The defect type information is extracted from the mask inspection reports. None of these defects are considered for defect avoidance.

  Equivalent XML parameter name: GlobalDefectTypeFilters.

  For example:

  ```
  global_defect_type_filters 1, 2, 1A, 2B
  <GlobalDefectTypeFilters>1, 2, 1A, 2B</ GlobalDefectTypeFilters>
  ```

- jd_chips_thresholds_file *filename*

  An optional parameter that filters chip data while processing a defect if the defect size is less than the threshold value specified in the job deck chips thresholds file. This parameter only applies to job decks.

  Typically, the boundary region (approximately 60 mm from the mask center) is a don’t care region. You can place large defects in that region. However, the boundary region is not completely empty, as there are chips with patterns present. To ensure that large defects are placed on these chips, use the jd_chips_thresholds_file parameter to filter the chip data.

  The format of the job deck chips thresholds file is:

  ```
  <chip_name, threshold_val>
  DEFAULT_DEFECTSIZE_THRESHOLD, <val>
  ```

  For example:

  ```
  Chip_A, 0.5
  Chip_B, 0.04
  Chip_C, 1.3
  DEFAULT_DEFECTSIZE_THRESHOLD, 0.1
  ```

In this example, for every solution identified in each major rotation, the tool checks if there are any exposed defects in Chip_A that exceed the 0.5 um threshold, any exposed defects in Chip_B that exceed the 0.04 um threshold, and any exposed defects in Chip_C that exceed 1.3 um. If any of these thresholds are exceeded, the solution should be discarded.

DEFAULT_DEFECTSIZE_THRESHOLD is an optional argument in the chips file that specifies the size of the threshold that can be applied to all other chips in the job deck. If this value is not specified, it defaults to 0.

Before running the Calibre MDPDefectAvoidance operation, the tool checks that chips listed in the text file exists in the job deck. If no chips are detected, a warning is output in the log file and the process continues.

For any chips that exist in the job deck but are neither mentioned explicitly in the text file, nor have DEFAULT_DEFECTSIZE_THRESHOLD specified, the tool defaults to the threshold size of 0.

The following apply:

- You can specify one, many, or none for the size, area, type, or chip-based defect size threshold filters.
- Defect filtering based on size, area, and type are combined by an OR operation.
- If any of size, area or type filter is satisfied, the defect is filtered and not considered for defect avoidance.

Equivalent XML parameter name: JdChipsThresholds file.

Output Parameters

You specify output directives in the parameters file for Calibre MDPDefectAvoidance.

- **dbfile *database_file_path***

  A required parameter that specifies the SQLite database file location containing information about blank mask, design and blank-design pairing.

  The SQLite database file should have either the extensions *.db* or *.sqlite*, which are case sensitive. For example, *test1.db* and *test2.sqlite* are SQLite database files.

  Equivalent XML parameter name: DatabaseFile.

- overwrite {true | false}

  An optional parameter that, if set to true and if a solution for blank design pairing already exists, overwrites the file.

  Equivalent XML parameter name: OverwriteDatabase.

- create_html_report {true | false}

  An optional parameter to instruct the tool to create HTML report. Specifying true enables the creation of the HTML report; specifying false deactivates report generation.

  Equivalent XML parameter name: CreateHtmlReport.

- html_report_windowsize *val*

  An optional parameter that defines the clip image size (in microns) inside the HTML report. The layout clip in the HTML report also considers defect size for clipping so that its window size can be equal to or greater than the size specified. For example, if the window size is specified as 6 microns and the defect width and height are 2 x 2 microns, the window clip in the HTML report is 8 x 8 microns. The default value is 6 microns.

  Equivalent XML parameter name: HtmlReportWindowSize.

Figure 3-44. HTML Report Window Size Node

```
<!--"CreateHtmlReport" node defines whether to create html report or not.
    Type: bool -->
<CreateHtmlReport>true</CreateHtmlReport>
<!--"HtmlReportSettings" node defines settings for html report such as HtmlReportWindowSize in microns.-->
<HtmlReportSettings>
 <!--"HtmlReportWindowSize" node defines window size (size of clip) of HTML report. Unit is in Microns-->
 <HtmlReportWindowSize>6</HtmlReportWindowSize>
</HtmlReportSettings>
<!--"CreateOasisReport" node defines whether to create oasis report or not.
```

- only_defect_marker {true | false}

  An optional parameter that generates defect markers in the visualization RDB file. When set to true, only one DefectMarker is reported in the visualization RDB file. If set to false, DefectMarker and DefectWithMargin markers are both reported in the RDB file. In the HTML report, only DefectMarker is generated in a layer, regardless of the parameter's value. The default value is false.

  Equivalent XML parameter name: OnlyDefectMarker.

Figure 3-45. OnlyDefectMarker Node

```
<!--"OutputInformation" node holds the output file information. -->
<OutputInformation>
 <!--"DatabaseFile" node specifies the database, in which the results are to be stored.
     Type: string- -->
 <DatabaseFile>/ina/inascratch/rsoni/DefectAvoidance/CustomerIssues/Samsung/57.DA_reverse_tone/New_Jobdeck/DA.db</DatabaseFile>
 <!--"OverwriteDatabase" parameter defines whether to overwrite the existing database or to create new
     database. Since the current algorithm works on single layout data and multiple blanks, one can overwrite the
     database on each run with different layouts.
     Type: bool -->
 <OverwriteDatabase>true</OverwriteDatabase>
 <!--"CreateHtmlReport" node defines whether to create html report or not.
     Type: bool -->
 <CreateHtmlReport>false</CreateHtmlReport>
 <!--"HtmlReportSettings" node defines settings for html report such as HtmlReportWindowSize in microns.-->
 <HtmlReportSettings>
  <!--"HtmlReportWindowSize" node defines window size (size of clip) of HTML report. Unit is in Microns-->
  <HtmlReportWindowSize>6</HtmlReportWindowSize>
 </HtmlReportSettings>
 <!--"OnlyDefectMarker" node defines whether to create DefectMarker only or DefectMarker and DefectWithMargin Marker both.

    Type: bool -->
 <OnlyDefectMarker>true</OnlyDefectMarker>
 <!--"CreateOasisReport" node defines whether to create oasis report or not.
     Type: bool -->
 <CreateOasisReport>false</CreateOasisReport>
 <!--"HtmlDirectory" node holds the absolute path of the report directory in which the html to be created.
     Type: string -->
 <ReportDirectory>/ina/inascratch/rsoni/DefectAvoidance/CustomerIssues/Samsung/57.DA_reverse_tone/New_Jobdeck/</ReportDirectory>
</OutputInformation>
```

- create_oasis_report {true | false}

  An optional parameter that instructs the tool to create an OASIS report. Specifying true enables the creation of the OASIS report; specifying false deactivates the report generation. The default value is false.

Equivalent XML parameter name: CreateOasisReport.

- **report_directory *report_dir_path***

  A required parameter that specifies the location where output report files and directories are stored. A unique report directory is generated within ***report_dir_path*** using a current time stamp. The results file, visualization results database file and HTML report-related files are stored inside that unique directory.

  Equivalent XML parameter name: ReportDirectory.

- sections_x *val* sections_y *val*

  Optional parameters that define the maximum parallelism that defect avoidance can achieve. The default value for sections_x and sections_y is 2 x 2. Parallel processing is limited by sections_x and sections_y and the number of cores available. The *val* is a positive nonzero number. The sections_x and sections_y parameters must be specified in separate lines in the parameter file.

  For example, comparing 1 x 1 for sections_x and sections_y with using 2 x 2 sections, a defect avoidance efficiency of 4 times can be achieved in an ideal case.

  Specifying sections_x and sections_y is best used when the defect avoidance process is run in Calibre MTflex mode.

  Equivalent XML parameter name: XSections for sections_x and YSections for sections_y.

Order of Priority for Defects and Layers

Defect classification (defect_type parameter) and layers priority (mask_data_layer_rule) together provides the order in which defects can be processed to find the solution space.

As illustrated in Figure 3-46, defect classifications H, M and L represents High, Medium and Low priority defects. In this figure, defects landing on Metal are more critical than defects landing on dummy fills, which are, in turn; more critical than defects appearing on OPC fills.

- The numbers on each cell represent the defects which fall in category represented by column and row. For example, “cell-1” means low priority defects appearing on OPC Fills.
- The arrows represent the order in which defects are to be ignored. For example, a solution is searched for by ignoring low priority defects appearing on OPC fills if a solution not found by considering all defects.
- The cells in green represent the starting point where all defects are considered to find the solution (find solutions without ignoring any defects or layers).
- The cells in red represent the most critical defects that cannot be ignored.

Figure 3-46. Defects Processing Order

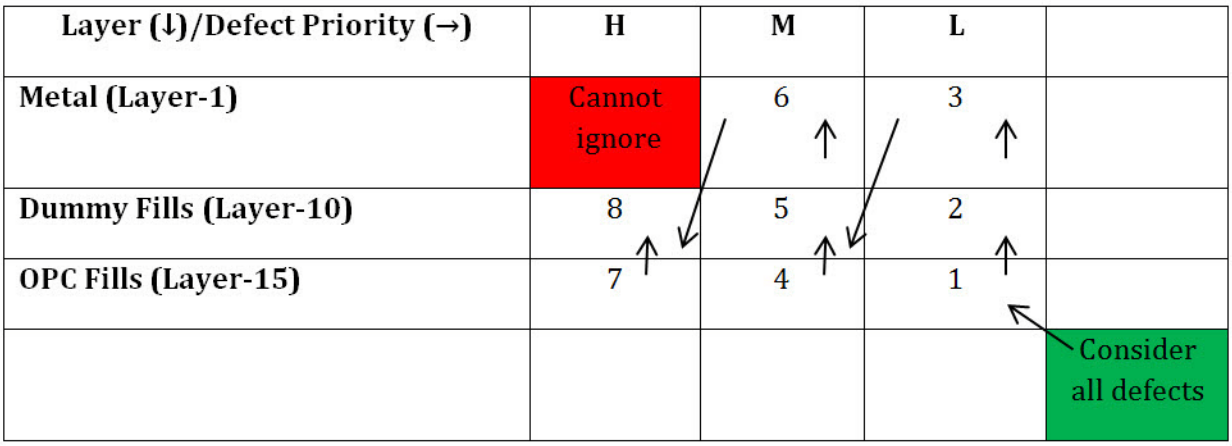


Figure 3-47 and Table 3-7 illustrate the order in which defects are ignored to find the solution space for Figure 3-46. The red circle represents high priority defects landing on most critical layer (Metal). If a solution is not available with step 9, the solution is reported with a minimum number of exposed defects. Exposed defects are of a high priority and overlap with the most critical geometries and cannot be ignored.

Figure 3-47. Order of Defects Processing to Find Solution Space

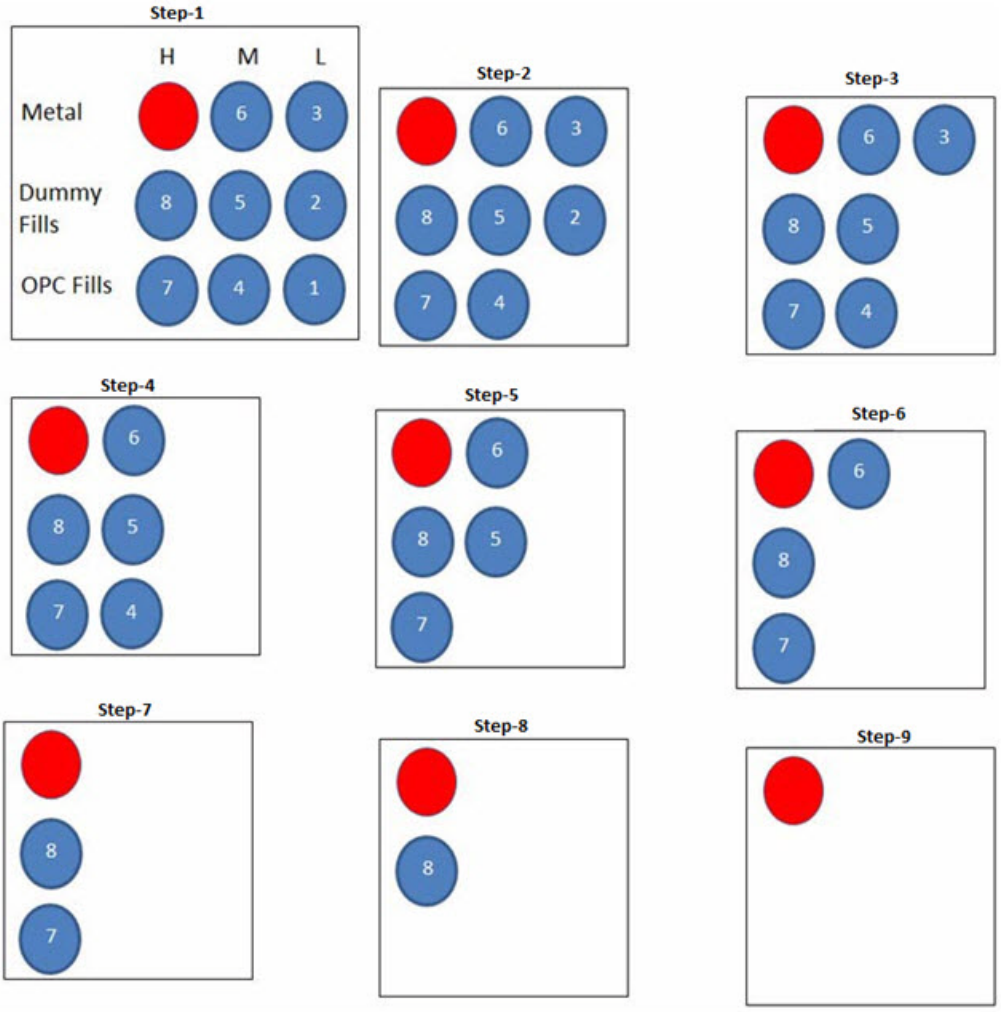


Table 3-7. Order of Defects Processing to Find Solution Space

Order	Description
Step-1	Search for a solution by considering all defects.
Step-2	If a solution is not found in Step-1, search for a solution by ignoring low priority defects from OPC Fills.
Step-3	If a solution is not found in Step-2, search for a solution by ignoring low priority defects from OPC Fills and Dummy Fills.
Step-4	If a solution is not found in Step-3, search for a solution by ignoring all low priority defects.

Table 3-7. Order of Defects Processing to Find Solution Space (cont.)

Order	Description
Step-5	If a solution is not found in Step-4, search for a solution by ignoring all low priority defects and medium priority defects from OPC Fills.
Step-6	If a solution is not found in Step-5, search for solution by ignoring all low priority defects and medium priority defects from OPC Fills and Dummy Fills.
Step-7	If a solution is not found in Step-6, search for solution by ignoring all low and medium priority defects.
Step-8	If a solution is not found in Step-7, search for solution by ignoring all low, medium priority defects and high priority defects from OPC Fills.
Step-9	If a solution is not found in Step-8, search for solution by ignoring all low, medium priority defects and high priority defects from OPC Fills and Dummy Fills. If a solution not found, then report the solution with an exposed defects count.

Note the following:

- If you do not specify defect priorities but specify layer priorities, then a solution is attempted with all layers; if no solution is found, defects on the OPC fills layer are ignored.
- If you do not specify layer priorities but specify defect priorities, then the tool first checks for a solution with all defects; if no solution is found, it ignores defects with Low priority.

Run Configuration File Format

Calibre MDPDefectAvoidance default run configuration file

Certain default behaviors and settings with Calibre MDPDefectAvoidance are controlled in an XML configuration file, *da_config.xml*. This includes debugging information, reticle radius and database unit information, and consideration of exposed defects, maximum solution width, and maximum exposed defects count to compute the usability factor for a blank-design pair.

You can find the configuration file *da_config.xml* in the following directory: *$MGC_HOME/pkgs/iccalibreda/pvt/calibredaconfig*. This file is read automatically when you run Calibre MDPDefectAvoidance and does not need to be explicitly loaded.

Format

This is an XML-format file, as illustrated in Figure 3-48.

Figure 3-48. Configurable Parameters File

```
<DAConfigFile version="5">
  <Debug>true</Debug>
  <LogLevel>3</LogLevel>
  <MaxAllowedDefectSizeInMicrons>10.0</MaxAllowedDefectSizeInMicrons>
  <ReticleRadius>76200</ReticleRadius>
  <WriterMinPrecision>true</WriterMinPrecision>
  <CalDbUnits>10000</CalDbUnits>
  <ConsiderExposedDefectsInUsabilityFactor>true</ConsiderExposedDefectsInUsabilityFactor>
  <MaxSolutionWidthInMicrons>1000</MaxSolutionWidthInMicrons>
  <MaxExposedDefects>100</MaxExposedDefects>
  <OasisReportDefectMarkerLayer>255</OasisReportDefectMarkerLayer>
  <DistributedModeDDE>false</DistributedModeDDE>
</DAConfigFile>
```

Parameters

- <Debug> {true | false} </Debug>

  An optional parameter that displays more detailed run information in a console window for debugging purposes. By default, Debug is set to false.

- <MaxAllowedDefectSizeInMicrons> *val* </MaxAllowedDefectSizeInMicrons >

  An optional parameter that specifies the maximum allowed defect size in microns. By default, MaxAllowedDefectSizeInMicrons is set to 10 microns. An error is issued if the blank inspection files contain defects with sizes larger than MaxAllowedDefectSizeInMicrons.

- <ReticleRadius> *val* </ReticleRadius>

  An optional parameter that aligns the center of the blank mask with center of the design. By default, it is set to 76200 microns (as shown in Figure 3-48).

- <CalDbUnits> *val* </CalDbUnits>

  An optional parameter that specifies database units. The default value is 10000 (as shown in Figure 3-48). Set the value of CalDbUnits by considering the maximum number of digits after the decimal point in the units used for the defect location, defect size from blank mask inspection report file, and user-specified setback value.

For example, if the number of places after the decimal point is 4, then value of CalDbUnits should be at least 10,000 (10^4). If the number of places after decimal point is 3, CalDbUnits should be at least 1000 (10^3).

- <WriterMinPrecision> {true | false} </WriterMinPrecision >

  An optional parameter that specifies whether or not the design's minimum precision should be computed automatically. The default value is set to false.

  If set to true, the minimum precision of the input pattern file or job deck is computed automatically. For job decks, the precision is considered as the Greatest Common Denominator (GCD) of all the chips' precisions, which also considers the placement information. For input files other than job decks, the precision is extracted from the layout information.

  If set to false, the tool considers the precision supplied through the CalDbUnits parameter in configuration file (which is 0.1 nm by default).

- <ConsiderExposedDefectsInUsabilityFactor> {true | false} </ ConsiderExposedDefectsInUsabilityFactor>

  An optional parameter that specifies whether or not the exposed defects count should be considered for the usability factor computation for a blank-design pair. The default value is set to true.

  When set to true, the exposed defects count is considered for usability factor computation. If set to false, the exposed defects count is not considered and the blank-design pair is considered not usable if the exposed defects count is greater than zero. In this case, the usability factor is set to zero.

- <MaxSolutionWidthInMicrons> *val*</MaxSolutionWidthInMicrons>

  An optional parameter in microns that specifies the maximum solution width to be considered while computing usability factor for a blank-design pair. By default, the value is set to 1000 microns.

- <MaxExposedDefects> *val*</MaxExposedDefects>

  An optional parameter that specifies the maximum exposed defects counts to be considered while computing usability factor for a blank-design pair. By default, the value is set to 100 defects. If the actual exposed defects count is greater than the MaxExposedDefects for a blank-design pair, then that pair is considered as an unusable pair and the usability factor is set to zero.

- <OasisReportDefectMarkerLayer > *val*</OasisReportDefectMarkerLayer>

  An optional non-negative parameter that specifies a defect marker layer in the OASIS report. The default value is 255.

- <DistributedModeDDE> {true | false} </DistributedModeDDE>

  An optional parameter that specifies whether or not Direct Data Entry (DDE) should be enabled when defect avoidance is run in Calibre MTflex mode. The default value is true. Values other than false are considered as true. The following also apply:

  - This is currently supported only with extended job deck format.
  - The Calibre MDPDefectAvodiance run should take less time when DDE is enabled.
  - Layout files should be accessible from remotes. If not accessible, then the remotes issue errors and the Calibre MDPDefectAvodiance run continues with the primary host.

- <LogLevel> {1 | 2 | 3} </LogLevel>

  An optional parameter that controls how logs are to be printed (for example, TIMERs in the primary log file). By default, LogLevel is set to 1. When you set LogLevel, you must also set Debug to true, as in the following example:

  ```
  <Debug>true</Debug>
  <LogLevel>3</LogLevel>
  ```

  Each LogLevel setting prints different timers.

  - LogLevel 1 prints the timers DESIGN_LOAD_TIME, PREOP_MASTER_TIME, DEFECTAVOIDANCE_TIME, EXPOSED_FINDER_TIME, HTML_REPORT_CREATION_TIME, and OASIS_WRITER_TIME in the primary log file.
  - LogLevel 2 prints all LogLevel 1 timers plus SECTION_PROCESSING_TIME in the primary log file.
  - LogLevel 3 prints all LogLevel 2 timers plus QUERY_TIME, INCREMENTAL_RESIZE_TIME, RESIZE_TIME, MERGE_TIME, NOT_TIME, and OVLP_TIME in the primary log file.

Pairing Optimizer Parameter File Format

Calibre MDPDefectAvoidance parameter file (used for the batch mode of the pairing optimizer)

The pairing optimizer parameters file is an XML file that specifies the options that are available in the pairing optimizer GUI. If the parameter file is not specified, the pairing run uses the current state of the database (depending on the availability of blanks and designs).

Format

This is an XML-format file, as illustrated in the following figure:

Figure 3-49. Pairing Optimizer Parameter File Format

```
<DAPairingParamFile version="2">
   <BlankInfo>
       <Blank name="demotxt.txt" status="AVAILABLE"/>
       <Blank name="demoldf.ldf" status="USED"/>
       <Blank name="demoklarf.klarf" status="AVAILABLE"/>
   </BlankInfo>
   <DesignInfo>
       <Design name="MASK1" status="AVAILABLE" blankCount="1"/>
       <Design name="MASK2" status="AVAILABLE" blankCount="1"/>
   </DesignInfo>
   <PairingInfo>
       <Pairing designName="MASK1" blankName="demotxt.txt" exclude="NO"/>
       <Pairing designName="MASK1" blankName="demoldf.ldf" exclude="NO"/>
       <Pairing designName="MASK1" blankName="demoklarf.klarf" exclude="NO"/>
       <Pairing designName="MASK2" blankName="demotxt.txt" exclude="NO"/>
       <Pairing designName="MASK2" blankName="demoldf.ldf" exclude="NO"/>
       <Pairing designName="MASK2" blankName="demoklarf.klarf" exclude="YES"/>
   </PairingInfo>
   <DatabaseInfo>
       <Database path="/home/demo/DA.db"/>
       <Database update="NO"/>
       <Database includeAllPairsFromDB="YES"/>
   </DatabaseInfo>
</DAPairingParamFile>
```

Parameters

Blank and Design Tables Parameters

These parameters specify which blank and designs are available or unavailable for pairing.

- name

  Corresponds to the mask blank name (mask_blank_name column) from the blank table or design name (mask_data_name column) from the design table.

- status

  Corresponds to the status column from the blank table or design table. If set to AVAILABLE, then the blank/design pair is available for pairing. If set to UNAVAILABLE, then the blank/design is not available for pairing. If set to USED, this indicates that the blank/design is already used and cannot be paired with any design/blank.

- blankCount

  Corresponds to the blank_count column from the design table. This attribute specifies the maximum number of blank masks with which the design should be paired.

Pairing Table Parameters

These parameters control the display of different columns in the pairing table.

- designName

  Corresponds to the design name (mask_data_name column) from the design table.

- blankName

  Corresponds to the mask blank name (mask_blank_name column) from the blank table.

- exclude

  Corresponds to the exclude column from the pairing table. If set to NO, then the designName and blankName pair is considered for pairing optimization. If set to YES, then the designName and blankName pair is excluded for pairing optimization.

Database Parameters

These parameters control whether or not the changes in the parameter file should be stored in the database and whether or not to include all the pairs from the database .

- path

  A required parameter that specifies the path to the database file containing all the information about the designs, blanks, and pairs specified in the pairing optimizer parameters file.

- update

  A required parameter that controls whether or not to update the original database. Valid values are YES and NO; the options are case-insensitive. If set to YES, then the original database is updated with the blank, design, and pairing status supplied in the parameter file. If set to NO, then the database remains unchanged.

- includeAllPairsFromDB

  A required parameter that controls whether or not to consider all pairs from the database. Valid values are YES and NO; the options are case-insensitive. If set to YES, then all pairs from the database are considered for optimization. If set to NO, then only pairs specified in the parameter file are considered for optimization.

  Note the following:

  - The blanks specified in the parameter file must be a subset of blanks available in database’s blank table.
  - The designs specified in the parameter file must be a subset of designs available in database’s design table.
  - The design-blank pairs specified in parameter file must be a subset of pairs available in the database’s pairing table.

Calibre MDPDefectAvoidance Environment Variables

Calibre MDPDefectAvoidance utilizes a number of environment variables to enable functionality to address specific issues for a defect avoidance run.

The following table summarizes the Calibre MDPDefectAvodiance environment variables described in this document.

Table 3-8. Calibre MDPDefectAvoidance Environment Variables

Variable	Description
CALIBRE_DA_HTML_REPORT_GENERATION_TIMEOUT *time_in_sec*	Specifies a positive non-zero integer value (in seconds) to set the timeout value. If report generation exceeds the time specified, the run is terminated. If report generation is aborted, any partial HTML report generated might be invalid.
CALIBRE_MDPDA_INCREMENTAL_RESIZE_ENABLE *val*	Prior to a defect avoidance run, each shape in a section is resized by a resize factor that incorporates defect size and any user specified setback values. As the defect size gets larger, the resize operation gets more costly. In such cases, this environment variable is used to break up the resize factor into smaller steps to speed processing. The step size can be set in *da_config.xml* using the IncrementalResizeFactorInMicrons parameter. The default is 0.01 um.
CALIBRE_MDPDA_OASIS_BINSIZE *val*	Changes the bin size used during bin injection of OASIS files to a user-specified value. The default value is 250 um.
CALIBRE_PLACEMENT_REPETITION_ENABLE 2112018	Stores an OASIS index more compactly. This addresses memory issues encountered while indexing OASIS files containing many repetitions of the same cell. Indexing time increases, but memory usage and index file size decrease significantly.
MDP_VBOASIS_INJECTION 1, MDP_VBOASIS_INJECTION_SIZE *val*, MDP_VBOASIS_INJECTION_PRESERVE 1	Three environment variables that speed up Calibre RVE HTML report generation can by using existing OASIS index and bin files. Set MDP_VBOASIS_INJECTION_SIZE to the bin size used when generating the bin files. The default value for MDP_VBOASIS_INJECTION_SIZE is 250 um.

Index

— O —

— P —

— Q —

— R —

— S —

— U —

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.