SIEMENS Calibre® Chip-Level Verification Quick Reference

Software Version 2021.2

Overview

Running a complete sign-off verification on a chip-level design that is under development can produce large numbers of results that are difficult to navigate, manage, and resolve. This document describes techniques and features available in Calibre that can help you analyze and manage DRC runs on incomplete data.

- Configuring a run to improve performance
- Reducing runtime with Calibre Reconnaissance
- Improving results debugging with DRC Analyze
- Configuring RVE for enhanced layer highlighting
- Excluding and waiving blocks from a verification run

Applying these features before sign-off verification provides you with a better starting point for debugging design verification issues.

Requirements

In addition to the licenses required by your rule file, you must have a Calibre RealTime Digital license to run Calibre nmDRC Reconnaissance and DRC Analyze.

No changes are necessary to your foundry rules or environment.

General Performance Tips

Setup Recommendations

- Run DRC using an OASIS layout with CBLOCK and Strict mode to achieve the best performance.
- For initial, standard Calibre nmDRC™ runs (without -analyze), set the DRC Maximum Results and DFM Defaults RDB Maximum statements to a reasonably small value, such as 500. Do not use ALL unless the design is small.
- Use the Layout Ignore Layer statement to completely exclude a layer from the run. This can be used to ignore fill shapes, but you can also use it to remove any incomplete or unnecessary layer from the verification run.

Single Machines

If there are enough CPUs and memory on a machine to handle the design and rule file complexity, consider using a single machine.

```
calibre -drc -hier -turbo -hyper "drc_rule_file"
```

Multiple Machines

```
calibre -drc -hier -turbo -hyper remote -remotedata
  -recoveroff -remotefile "remote_file" "drc_rules"
```

Always use the -remotedata and -recoveroff options for jobs with remotes. After the initial hyper-flex run, review the transcript for "SHARED = " for the final operation. If the value is not equal to "0/0", and there is free memory on remotes, then specify "-remotedata 2" in the next Calibre invocation.

For an *N* CPU remote, Calibre uses an additional *N* GB of memory for the data server, which avoids transferring data amongst the master and remotes.

Runs on Small Blocks

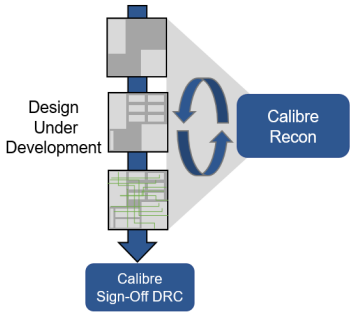
- Run multi-threaded on a single machine. For improved performance, use as many CPUs as available.
- Always enable hyperscaling.

Runs on Large or Top-Level Blocks

- Use the MTflex option (distributed run).
- Use HyperRemote to execute more operations in parallel (the minimum recommended number of CPUs is 24).
- Increase the number of CPUs for large designs.

Calibre Reconnaissance

Calibre Reconnaissance can significantly improve performance for non-sign-off verification runs. You should run Calibre nmLVS Recon SI and Calibre nmDRC Recon during the development of your design and for the initial runs on large, finalized designs. Once you have corrected issues found with the Calibre Recon runs, you can proceed to a full Calibre sign-off run.



The results from a Calibre Recon run are similar to a normal run, so you can follow the same debugging and fixing flow that you use in your established verification process.

Calibre nmDRC Recon — The Calibre nmDRC Reconnaissance tool performs automated rule check selection that provides fast feedback to help designers identify systemic design issues that may result in many errors. A Calibre nmDRC Recon run primarily excludes context-dependent rules, such as connectivity, delta v, and multi-patterning rule checks, that are likely to result in long run times and are unlikely to provide actionable results for an unfinished layout.

The excluded rule checks require significant context awareness and can generate hundreds or thousands of violations at different locations across the chip; usually such violations are resolved indirectly as the design matures. By contrast, Calibre nmDRC Recon rules are fast and identify results that are typically geographically close to the source of the design flaw. The faster and more targeted Calibre nmDRC Recon run enables you to identify and fix significant issues in your layout before performing sign-off verification runs. Invoke Calibre nmDRC Recon as follows:

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

If you want to run only the rule checks that were unselected, use the inverse option:

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

To add or remove rules from a Calibre nmDRC Recon run, include the DRC Recon Add Check, DRC Recon Add Check By Layer, DRC Recon Remove Check, and DRC Recon Remove Check By Layer statements in your rule file.

To run Calibre nmDRC Recon from Calibre Interactive, enable the "Recon" checkbox on the **Inputs** tab. To run from Calibre RealTime Digital, select a check recipe for Recon.

Calibre nmLVS Recon SI — Performs stand-alone short isolation independent of an LVS run. Such runs are typically very fast and can be used early in the design verification process to avoid various problems in later LVS runs. Invoke Calibre Recon nmLVS™ as follows:

Rule File Flow

```
calibre -recon -si['=' { ALL | IO | PG | tcl_script }]
   [opts] rule_file_name
```

Mask SVDB Directory Flow

```
calibre -recon -si['=' { ALL | IO | PG | tcl_script }]
   [opts] -svdb svdb_dir [top_cell]
```

Calibre DRC Analyze

The DRC Analyze feature uses Calibre RVE to present various statistical representations of DRC results. You can use these enhanced results to focus on chip-level errors that have a higher fixing priority and impact on the design flow. The analysis results include histograms, colormaps, and sortable tables of rule checks.

The histograms can be based on hierarchical cells or windows. You can customize the histograms by selecting different results and windows under different bin ranges.

The colormaps can be viewed in separate windows or overlayed on the layout. The colormaps indicate the distribution of errors across a design. You can use the colormap as a navigational starting point to investigate the cause of different rule checks and then probe down into the per-cell and per-window errors in the design.

A key advantage of this flow is that it does not require any modifications to the foundry rules and the overhead on runtime is small (approximately 10% increase in runtime and 20% increase in memory).

Running Calibre Recon and DRC Analyze

1. Invoke Calibre nmDRC Recon with DRC Analyze mode:

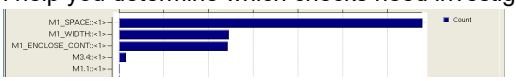
   ```
   calibre -drc -hier -turbo -analyze -recon rules.svrf
   ```

2. Open the layout in your preferred layout viewer and open the *analyze.dfmd* database in Calibre RVE for DFM. For example:

   ```
   calibredrv -m chip.oas -rve -dfm analyze.dfmdb
   ```

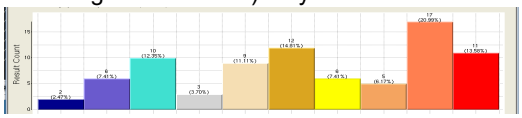
3. Choose **View > Bar Chart** from the **Chip Summary** tab.

   This provides a graphical distribution of the rule check results. This can help you determine which checks need investigating.

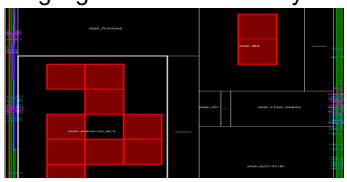


4. Choose **View > Table** to return to the previous view.
5. Right-click on a rule check and choose **Histogram Hierarchy Windows > Count**.

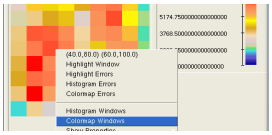
   This window reports the number of errors found within windows of your design. The windows with high numbers of errors (the bars on the right of the chart) may be of the most interest.



6. Right click on the bar with the most results and choose **Highlight this bar**. This highlights the windows in your layout.



7. In the **Chip Summary** tab, right-click on a rule check and choose **Colormap Hierarchy Windows > Count**. This colormap displays the location of the violations from the selected check.

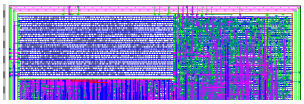


8. Right-click on the colormap and choose **Colormap Windows**.
9. This overlays the colormap on your layout window.
10. Right-click on one of the windows in the colormap in Calibre RVE and choose **Highlight Errors** This operation overlays the actual errors on your layout.

Tip: You can also create an on-the-fly RDB for a specific rule by right-clicking on the rule and choosing **Browse Hierarchy Errors > Count** in the **Chip Summary** tab.

Excluding Incomplete Blocks (Gray Box)

When performing initial verification on the top-level, a design may contain unfinished blocks, even though the connections from the blocks to surrounding cells are established. The top-left block in this example is unfinished, but the external connections are routed.



A standard DRC run may include numerous results from the incomplete blocks. This increases difficulty in debugging real problems in the design. You can exclude blocks from a run, yet still check the interacting regions on the perimeters of the blocks using the Calibre Auto-Waivers tool or an SVRF statement.

Gray Box Using DRC (SVRF)

```
LAYOUT WINDEL CELL cell_name ... [HIER] {[BY LAYER layer
[PRIMARY]] | [ORIGINAL [OCCUPIED]]} [HALO distance]
```

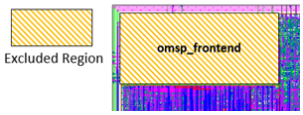
The following command removes the contents of the omsp_frontend cell from the verification run, but keeps 100 user units of geometry around the extents of the cell:

```
Layout Windel Cell "omsp_frontend" HIER HALO 100
```

Gray Box Using Calibre Auto-Waivers

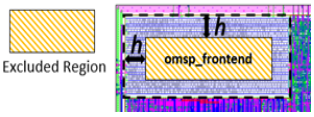
Include the following command in your criteria file to remove blocks from the verification run with waivers:

```
EXCLUDE_CELL cell_name [cell_name …] [HALO halo_value]
  [BY_LAYER layer_name] [PRESERVE layer_to_keep …]
```



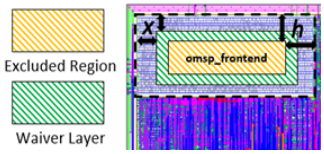
Remove the entire block from processing:

```
EXCLUDE_CELL omsp_frontend
```



Remove the block from processing, but keep the distance *h* from the edges and keep layers M1 and V1..

```
EXCLUDE_CELL omsp_frontend
  HALO h PRESERVE M1 V1
```



Remove the block from processing, but keep the distance *h* from the edge of the block. Waive all errors for ruleA within the waiver layer, which is *x* distance from the cell extents .

```
EXCLUDE_CELL omsp_frontend
  HALO h
  ruleA WAIVE_EXTENT x
```

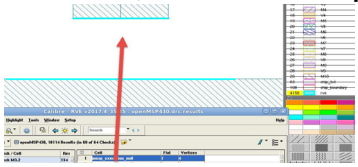
Note: INCLUDE_CELL can be used to *only* include certain blocks.

Optimizing Result Highlighting

Calibre RVE supports optional highlighting in the DRC results tab that enables you to easily view design polygons involved in the rule check results, without manually setting visibility on the layers.

To use this feature, you must create a CTO file that controls Calibre RVE layer visibility in the design tool. Do the following:

1. In Calibre Interactive, click **Outputs** and enable the “Create CTO file” and “Groups” options.
2. Click **Run DRC**.
3. In Calibre RVE, choose **Tools > Import CTO File**.



Layers not involved in the check are automatically hidden upon highlighting the result

Note, you can also create a CTO file from Calibre RVE by choosing **Tools > Create CTO File** from a DRC results tab.

SIEMENS