SIEMENS EDA

Calibre® Post-Tapeout Flow User's Manual

Software Version 2021.2

SIEMENS

Unpublished work. © 2021 Siemens

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

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

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

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

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

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

Table of Contents

List of Figures

List of Tables

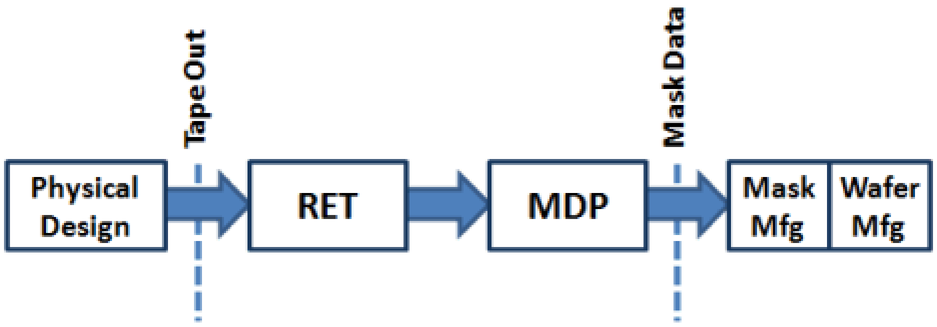
Chapter 1
Introduction to Post-Tapeout Flows

This chapter provides background information about the Calibre post-tapeout flow.

What is Calibre Post-Tapeout Flow?

Calibre® post-tapeout (PTO) flow is a sequence of data processing operations between design tapeout and mask making. It extracts the relevant layout information from an original design database, applies manufacturing-driven modifications and verifications, and generates a single-layer database that is ready for manufacturing. The post-tapeout operations are controlled by a common command language (SVRF/TVF) and can either be integrated into a single continuous run or be separate and sequential runs.

Figure 1-1. Calibre Post-Tapeout Flow



Usually, a post-tapeout flow includes Optical and Process Correction (OPC), scattering bar generation, Mask Data Preparation (MDP), and similar tasks. These steps occur after a design has been verified as DRC and LVS clean, backannotation and simulation have been performed, and the design (or a subset of layers) has been approved for tapeout.

This manual discusses functions that support specific applications in the post-tapeout flow, which are not recommended as the standard settings for general Calibre applications. In particular flows and applications, certain of these functions have demonstrated substantial runtime improvements due to improved Calibre scalability. In many of these cases, the higher scalability allows further reduction in runtime by adding more compute resources. The output of these functions yield a lithographically equivalent result to the output of a standard hierarchical

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Calibre run, with a less-intuitive database representation compared to the original design database.

Before using these operations, Siemens EDA encourages you to analyze carefully the scope of the operations and the possible negative implications if they are not used under the correct circumstances. Extensive testing in your production environment is highly recommended.

Note

This document captures the current state of known benefits, along with the known drawbacks, of using these functions. Results have been derived from analysis of empirical data. In most cases, there is no way to predict what benefit you may see for a particular design.

Terminology

This section includes definitions of some key terms that are frequently used when describing the usage of Calibre in a post-tapeout flow.

Runtime

The elapsed real time for completion of a Calibre job is a primary metric in production flows. Some runtime factors are the amount of computation work, hardware speed, number of processors, structure of the task (such as layout cells), dependencies between tasks, disk access time, memory swapping, and network transfer delay.

Scaling

Scaling is the effective number of active processors, calculated as the total CPU time from all processors divided by the elapsed real time. It is commonly used as an indicator of the performance of parallel processing with MT and Calibre® MTflex™. Scaling can be calculated over a whole Calibre job, or over smaller increments, such as particular operations. It is useful to consider scaling separately for phases of the job which are more or less parallel. For example, file loading and parsing are much less parallel.

Some major factors that influence scaling are the number of processors, the number of threads created from partitioning of the tasks, file I/O, competing jobs, and hardware failure.

Scaling is reduced by the time taken for data transfer. Some data transfer time is always required for the parallel sub-processes, as well as the full job. Thus, the observed scaling value is always less than the number of available processors. For example, if you have 32 processors available for a job, a scaling value of less than 32 will be observed.

If the processors or network are running other jobs at the same time, the observed scaling is reduced and more variable.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

When a remote processor fails or becomes unavailable, it does not contribute to scaling, so the scaling effect is reduced accordingly.

Most processing jobs are very complex, containing various cell sizes with processing dependencies. For parallel processing, the job must be divided into independent chunks of data. The ability to partition the processing work into a sufficient number of parallel equal-size chunks is highly dependent on the structure of the design and the operations. Thus, scaling will be significantly less than expected in many cases. The complexity of these factors make run time and scaling highly variable, and difficult to predict.

Scalability

Scalability is the effectiveness of multiple processors to reduce the real time of a Calibre job. It is another way to express scaling, normalized to the number of available processors. It is typically expressed as a percentage. Generally for Calibre MTflex, only the number of remote processors is used in this calculation, because the main task of the primary processors is to serve data to the remote processors, which perform the bulk of the calculations. Scalability indicates the efficiency of the compute portion of the job, and the ability to utilize additional parallel processors to share the workload. Scalability generally decreases as more processors are added, so it only indicates the current operating state, and helps bound the expected runtime with more or fewer processors.

Hierarchical Efficiency

Hierarchical designs are composed of shapes within cells, and cells placed as instances in higher-level cells. Hierarchical efficiency is observed when there are multiple instances of a cell, which cause the total file size to be smaller than the flat layout.

This hierarchical efficiency is the basis for reduction in processing time and memory of a hierarchical layout with hierarchical Calibre applications. Hierarchical efficiency is commonly measured by the flat to hierarchical geometry count (FGC/HGC) or the flat to hierarchical edge count (FEC/HEC). (See the "Layer Statistics in Hierarchical Processing" section of the *Calibre Verification User Manual*.) The partitioning of a design into cells has benefits to parallel processing even when cells are not instantiated more than once. However, not all hierarchical structures are efficient, and hierarchical efficiency does not always lead to efficient processing. For example, overlap of cells can cause dependencies that complicate the processing of the cells independently.

Hierarchy Degradation

During the course of geometric processing, there are typically factors that degrade the efficiency of the hierarchy. One example of this is interaction of shapes between cell instances. In some cases, the results naturally move upward in the hierarchy. This may degrade the hierarchical efficiency and increase the processing time for similar operations following.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Syntax Conventions

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

Table 1-1. Syntax Conventions

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

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Chapter 2
Calibre Post-Tapeout Tools

This chapter lists the various products by categories, and links to their manuals. It also provides recommendations for settings common across the flow such as invocation switches and required SVRF statements.

Common Tools in the PTO Flow

The Calibre family includes a complete set of tools for the generation and verification operations from tapeout to mask write. The following tools are commonly used in the post-tapeout flow.

Calibre PTO Generation Commands

- OPCBIAS in the *Calibre Rule-Based OPC User's and Reference Manual*.
- LITHO NMBIAS in the *Calibre Rule-Based OPC User's and Reference Manual*.
- OPCSBAR in the *Calibre OPCsbar User's and Reference Manual*.
- LITHO OPC in the *Calibre OPCpro User's and Reference Manual*.
- LITHO DENSEOPC in the *Calibre nmOPC User's and Reference Manual*.
- RET NMDPC in the *Calibre Multi-Patterning User's and Reference Manual*.
- RET SBAR in the *Calibre nmSRAF User's and Reference Manual*.
- RET MBSRAF in the *Calibre nmSRAF User's and Reference Manual*.
- cnSRAF in the *Calibre nmSRAF User's and Reference Manual*.
- MDP EMBED in the *Calibre Mask Data Preparation User's and Reference Manual*.
- MDP MAPSIZE in the *Calibre Mask Data Preparation User's and Reference Manual*.
- LITHO MASKOPT (formerly MDP MASKOPT) in the *Calibre Mask Data Preparation User's and Reference Manual*.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- MDP OASIS_EXTENT in the *Calibre Mask Data Preparation User's and Reference Manual*.
- MDPMERGE in the *Calibre Mask Data Preparation User's and Reference Manual*.
- LITHO MPC in the *Calibre nmMPC and Calibre nmCLMPC User's and Reference Manual*.
- FRACTURE in the *Calibre Mask Data Preparation User's and Reference Manual*.
- General SVRF commands in the *Standard Verification Rule Format (SVRF) Manual*.

Calibre PTO Verification Commands

- LITHO ORC in the *Calibre OPCpro User's and Reference Manual*.
- LITHO OPCVERIFY in the *Calibre OPCverify User's and Reference Manual*.
- LITHO PRINTIMAGE in the *Calibre OPCpro User's and Reference Manual*.
- MDPVERIFY in the *Calibre Mask Data Preparation User's and Reference Manual*.
- MDPSTAT in the *Calibre Mask Data Preparation User's and Reference Manual*.

Viewers

The layout viewers share common functionality, which is documented in the *Calibre DESIGNrev Layout Viewer User's Manual*.

- DESIGNrev in the *Calibre DESIGNrev Layout Viewer User's Manual*.
- LITHOview in the *Calibre WORKbench User's and Reference Manual*.
- WORKbench in the *Calibre WORKbench User's and Reference Manual*.
- MDPview in the *Calibre MDPview User's and Reference Manual*.
- Fracture GUI in the *Calibre MDPview User's and Reference Manual*.
- RET Flow Tool in the *Calibre WORKbench: RET Flow Tool User's Manual*.

The command reference for scripting the viewers is contained in the *Calibre DESIGNrev Reference Manual: Batch Command Support for Calibre DESIGNrev, Calibre WORKbench, Calibre LITHOview, and Calibre MDPview*.

Converters and Utilities

- filemerge in the *Calibre DESIGNrev Reference Manual*.
- peek in the *Calibre DESIGNrev Reference Manual*.
- gds2oasis in the *Calibre for the Open Artwork System Interchange Format (OASIS)* manual.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- mebes2oasis in the *Calibre MDPview User's and Reference Manual*.
- jeol2oasis in the *Calibre MDPview User's and Reference Manual*.
- vsb2oasis in the *Calibre MDPview User's and Reference Manual*.
- hitachi2oasis in the *Calibre MDPview User's and Reference Manual*.
- micronic2oasis in the *Calibre MDPview User's and Reference Manual*.
- jobToOasis in the *Calibre MDPview User's and Reference Manual*.
- jeolTrapStats in the *Calibre MDPview User's and Reference Manual*.
- hitachiTrapStats in the *Calibre MDPview User's and Reference Manual*.
- micronicTrapStats in the *Calibre MDPview User's and Reference Manual*.
- vsbTrapStats in the *Calibre MDPview User's and Reference Manual*.

Metrology

- Test pattern generation in the *Calibre WORKbench User's and Reference Manual*.
- Calibre Metrology Interface (CMi) in the *Calibre Metrology Interface (CMi) User's Manual*.
- Calibre Metrology API (MAPI) in the *Calibre Metrology API (MAPI) User's and Reference Manual*.

Mask and Wafer Inspection and Correction

- Calibre DefectReview (formerly Calibre MDPDefectReview) in the *Calibre DefectReview User's Manual*.
- Calibre DefectClassify (formerly Calibre MDPPatternClassify) in the *Calibre DefectClassify User's Manual*.
- Calibre MDPAutoClassify in the *Calibre MDPAutoClassify User's Manual*.
- Calibre MDPDefectAvoidance in the *Calibre MDPDefectAvoidance User's Manual*.

Control, Monitor, and Analysis

- Cluster Manager (CalCM) in the *Calibre Cluster Manager (CalCM) User's Manual*.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Common Calibre Settings and Modes for PTO Applications

Post-tapeout applications use settings that are uncommon in physical verification applications. This section summarizes the important options that differ once a design is complete.

Command Line Options

The command line options are used when invoking a Calibre run. This section does not provide complete reference information; it is advice for typical post-tapeout situations only.

Command line options are listed in Table 2-1. They are documented in the "Calibre nmDRC and Calibre nmDRC-H Command Line" section of the *Calibre Verification User's Manual*.

A typical command line is

```
$CALIBRE_HOME/bin/calibre -drc -hier \
   -turbo -remotefile remote.cfg -hyper myjob.svrf |& tee myjob.log
```

Table 2-1. Calibre Command-Line Options

Option	Additional Notes
-drc -hier or -pto	Use either traditional hierarchical or Calibre® FullScale™ engines.
-turbo (no value), -turbo_litho (no value), -remotefile *file*	Calibre® MTflex™ is very commonly used in the PTO flow to reduce runtime by parallel processing.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 2-1. Calibre Command-Line Options (cont.)

Option	Additional Notes
-remotedata	The -remotedata switch invokes the Remote Data Server (RDS). RDS is useful in some PTO environments where the runtime or functionality is limited by the hardware: • where memory on the primary host is not large enough for the job. (LVHEAP is larger than physical memory; swap used.) • where runtime is limited by network speed. (A common example: 1 GB Ethernet with more than 200 remote CPU cores.) Where possible, increase the memory and network bandwidth rather than use RDS. Other considerations in the use of RDS include: • RDS increases memory requirements on the remote nodes; typically 2 GB per 4 remote CPU cores, or 1 GB with the -recoveroff option. • RDS requires more reliability of the remote hosts. Remote failures with RDS can cause the whole job to fail. • RDS can make large or long-running jobs more susceptible to network problems. • RDS is not supported for MT.
-recoverremote -recoveroff (default)	(RDS only) These options control recovery from remote host failure. By default, RDS does not use -recoverremote. The default, -recoveroff, disables recovery. Disabling recovery halves the RDS remote memory consumption, from a typical 2 GB per four CPU cores to 1 GB per four CPU cores. Disabling recovery also slightly improves run time on *reliable* clusters. (If any remote fails with -recoveroff, the entire run fails.) Generally, in PTO flows with long runs on large clusters, the -recoverremote option is recommended. Only use -recoveroff if there is high confidence that all remotes will not fail.
-hyper	Hyperscaling is useful in some PTO jobs to improve scaling. It increases memory requirements on the primary and remote nodes. Hyperscaling cannot be used with the Calibre FullScale or Calibre Distributed Flat platforms.
-hyper remote	Hyper remote is typically not recommended for PTO jobs, but may be useful in some cases to further improve scaling. Hyper remote increases memory requirements on the remote nodes. Remotedata is required when using hyper remote.
-64	Only required in versions prior to 2011.1.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 2-1. Calibre Command-Line Options (cont.)

Option	Additional Notes
|& tee myjob.log or >& myjob.log	It is strongly recommended that you routinely capture the transcript. This can be done with a redirect (>&) or by piping the output through "tee". The tee utility, which captures terminal output, is available on most Linux platforms. Unlike redirects, it allows you to see the run's progress also in the terminal window. References for interpreting the Calibre transcript are in the "Interpreting the Calibre Transcript for PTO Applications" chapter.

Related Topics

Calibre Data Construction

Remotefile Contents

The lines following represent a typical remotefile for PTO use. The MONITOR commands are recommended for good indication of performance and diagnosis of problems.

This is a basic remote file for two hosts:

```
REMOTE HOST remote1
REMOTE HOST remote2
```

The next example is also a basic remote file, but launches the remote hosts faster because it uses concurrency:

```
LAUNCH CLUSTER COUNT 2
REMOTE COMMAND launch_remote.csh //Must provide the remote shell script
                                 //to start rcalibre on each remote.
```

The remote shell script (*launch_remote.csh* in the example above) should be like the following:

```
#!/bin/csh -f
rsh remote1 $CALIBRE_HOME/bin/rcalibre /tmp 1 MGC_HOME $CALIBRE_HOME \
   -mtflex $CALIBRE_REMOTE_CONNECTION -v &
rsh remote2 $CALIBRE_HOME/bin/rcalibre /tmp 1 MGC_HOME $CALIBRE_HOME \
   -mtflex $CALIBRE_REMOTE_CONNECTION -v &
```

The following reporting statements are recommended in the remote file:

```
MONITOR SYSTEM REPORT STDOUT  //Or REPORT ASCII -- STDOUT makes
                              //transcripts larger but easier to find and
                              //facilitates support.

MONITOR SYSTEM SWAP 8000 KILLONE PRINT //Recommended to be 25% of the
                                       //remote memory size. Should be
                                       //used instead of MAXMEM to handle
                                       //swapping.
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

```
MONITOR SYSTEM HEARTBEAT 4 PRINT    //Recommended to reduce the recovery
                                    //time for unresponsive remotes.

MONITOR LOCAL LOAD INTERVAL 300
MONITOR REMOTE MEMORY INTERVAL 300 //Best for debug use only.
                                   //Gives more memory detail in the
                                   //remote logs.
```

Tip

When debugging, change the location of remote logs to a shared disk accessible by the primary host. The directory is specified by the DIR option of REMOTE HOST, or the *rundir* argument to rcalibre.

The choice of report destination for MONITOR SYSTEM depends on your analysis methods and preference. REPORT STDOUT interleaves the additional monitoring results into the main Calibre transcript. REPORT ASCII creates a separate file, resulting in a cleaner main transcript but requiring extra effort for coordinated analysis and file transfer.

Related Topics

Configuration File Reference Dictionary [Calibre Administrator's Guide]

Calibre Settings in the Rule File

The following specification statements are commonly used in SVRF rule files for PTO jobs.

Table 2-2. Common SVRF for PTO Rule Files

Statement	Comment
DRC MAXIMUM RESULTS ALL	For PTO layer generation operations. All is not on by default.
DRC RESULTS DATABASE file OASIS PSEUDO STRICT CBLOCK	For best runtime in the PTO flow, use PSEUDO with the OASIS®[1] format. STRICT mode facilitates best performance for vboasis_path input to MDP Embed. CBLOCK mode minimizes the size of the layout output file.
LAYOUT MAGNIFY AUTO	Handles production flow with various possible input layout precisions.
LAYOUT INPUT EXCEPTION SEVERITY PRECISION_INPUT_FILE 0	Handles changes of input precision.
LAYOUT INPUT EXCEPTION SEVERITY PRECISION_RULE_FILE 0	Handles changes of input precision.
LAYOUT INPUT EXCEPTION SEVERITY DUPLICATE_CELL	For assembling multiple layout files.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 2-2. Common SVRF for PTO Rule Files (cont.)

Statement	Comment
LAYOUT BASE LAYER L1 L2	Improves execution time. See "Optimal Performance Using LAYOUT BASE LAYER" on page 22.
LAYOUT TURBO FLEX YES	Typically used for marginal incoming hierarchy.
LAYOUT ULTRA FLEX YES	Typically used for very poor incoming hierarchy.

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.

These rules are common but should *not* be used:

Table 2-3. Deprecated Common SVRF for PTO Rule Files

Deprecated Statement	Comment
DRC MAXIMUM VERTEX	For best runtime in most post-tapeout situations, this statement is not recommended. Its default value of 4096 vertices is usually sufficient.
LAYER DIRECTORY	This statement is not recommended.
PUSH	The Push command is no longer generally recommended for reducing runtime. An exception is for dummy fill layers that may be passed whole or in part to Litho or RET operations. In these cases there may be some runtime benefit from doing a Push on the input fill layer to the target layer. Always evaluate the benefit to the overall runtime.

OASIS Layout Format in the PTO Flow

The OASIS layout format is commonly used in the PTO flow for reduced file sizes and best runtime performance.

Some types of Calibre jobs create additional files such as index files when processing an OASIS layout. It is recommended that you create a symbolic link (soft link) in your file system in an unshared directory that points to the OASIS file rather than using the centrally stored file. This best practice prevents file conflicts when multiple jobs are using the same layout.

OASIS format and application are described in the *Calibre for the Open Artwork System Interchange Standard (OASIS)* manual.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Chapter 3
Calibre Data Construction

Calibre offers several platforms for post-tapeout environments, which each handle hierarchical layout data differently. This chapter provides an overview of the choices and advice on optimizing run time.

There is a hierarchy of choices to make: platform, construction method, processing mode, and data paths.

"Platform" refers to the core processing executable. It is controlled by the invocation switches.

- **calibre -drc -hier —** This is the traditional Calibre hierarchical platform, and the only one that is also used pre-tapeout. There are several parts that can be optimized for typical post-tapeout conditions, but its primary design serves physical verification needs.
- **calibre -drc -flat —** This invokes Calibre Distributed Flat, a hybrid approach. As of version 2018.1, this platform is deprecated.
- **calibre -pto —** This runs the Calibre FullScale engine, which has been designed for very large layout databases with shallow, degraded hierarchy. It scales better than the Calibre hierarchical platform. The Calibre FullScale platform uses a different parallelization algorithm designed for post-tapeout conditions but still makes use of the hierarchical platform's Calibre MTflex and remote data server functionality.

Multi-processing requires the Calibre hierarchical engine, Calibre Distributed Flat, or the Calibre FullScale platform. The breakdown of the hierarchical layout data for multi-processing is determined by the construction method.

"Construction method" refers to how the data hierarchy of the design is represented in Calibre memory. Much of this is documented in the "Distributed Calibre" chapters of the *Calibre Administrator's Guide*. This chapter provides additional details on the methods typically used for post-tapeout runs: TURBOflex, ULTRAflex, and MONDOflex. The construction method is determined by a combination of invocation switches, SVRF statements, and environment variables.

"Processing mode" is specific to post-tapeout operations that use a setup file. The term refers to how data inside cells is processed, and is controlled by the processing_mode command.

"Data paths" refer to the choices on how results are passed between Calibre operations. Most post-tapeout rule files include several sequential operations. Results from one operation can be written to a central file, a shared memory location, or streamed directly into the next operation. There are several factors that influence the optimal choice for your computing environment.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Note
For efficient hierarchical processing with Calibre Hierarchical or Calibre Distributed Flat platforms, you must first properly configure Layout Base Layer, as described in "Optimal Performance Using LAYOUT BASE LAYER" and "How to Tell Which Layers are Base Layers".

Optimal Performance Using LAYOUT BASE LAYER

To obtain the best performance from the Calibre Hierarchical Engine, it is recommended that you supply an optional statement, LAYOUT BASE LAYER, to identify the layout base layers. LAYOUT TOP LAYER, the alternative to LAYOUT BASE LAYER, is not recommended.

When to Use

- For hierarchical designs.
- When processing more than one layer.

Why to Use

When Calibre's hierarchical database constructor reconstructs (or injects) the hierarchy, it represents the hierarchy structure in such a way as to maximize performance for layer operations. However, when multiple layers are involved, the optimum hierarchical structure could vary from layer to layer. In general, those cells that contain the device level layers tend to be the most repeated structures and upper level metal layers tend to form unique structures. Therefore, a hierarchical structure that fits cells containing device layers is most likely to provide the best overall performance.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

For the database constructor to calculate the best overall hierarchy, it must make distinctions between two types of cells, base-layer cells (device) and top-layer cells (non-device forming). It recognizes the cell type based on the presence or absence of two types of layers:

- **Layout base layers** — Containing data configurations that are likely to be replicated in multiple places on the mask, such as the layers in an arrayed bit cell. Layout base layers typically include poly, diffusion, implant, and contact (not via) layers.
- **Layout top layers** — Containing data configurations that are unique (not replicated elsewhere on the mask). This typically includes physical layers such as metal, via (not contact), bump, pad, and fuse, plus special purpose layers such as markers, artificial cell boundaries, and text.

Note

The term "layer" refers to a "drawn layer" in the layout database, not the photomask layer, which may require multiple drawn layers to produce.

The performance of Calibre can be severely degraded if no base layers are designated, if the base layers do not represent an efficient hierarchy, or if they represent conflicting hierarchy.

The Difference Between the Two Statements

You identify layout base layers using either LAYOUT BASE LAYER or LAYOUT TOP LAYER. In most cases, using LAYOUT BASE LAYER is preferred.

- If you use LAYOUT BASE LAYER, the Calibre database constructor fits the hierarchy only to those cells that contain layout base layers. Note that if you use LAYOUT BASE LAYER, Calibre disregards LAYOUT TOP LAYER statements it encounters.
- If you use LAYOUT TOP LAYER alone, the Calibre database constructor fits the hierarchy to all cells except those that contain only layout top layers. When using LAYOUT TOP LAYER, failure to list all layout top layers could negatively impact performance.

How to Tell Which Layers are Base Layers

When performing RET or MDP, it is not always easy to distinguish between layout base layers and layout top layers. If you are processing more than one input layer of a hierarchical design, use the following guidelines.

- If the rule file contains at least one LITHO DENSEOPC operation, use the calflowoptimize utility to create a LAYOUT BASE LAYER statement. The input file can be in TVF or SVRF, but must contain no undefined environment variables or obsolete operations.

  ```
  calflowoptimize input_file [-o output_file]
  ```

  The default output file is *dummy.svrf*.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- Examine the rule files used for DRC or LVS for guidance from base layers used earlier in the flow.
- Examine some input layouts to determine which layers are used to form devices with efficient hierarchy.
- If the target photomask is on or below the contact layer, then other photomask layers that are on or below contact should be considered for base layers. Other layers above contact should NOT be base layers.
- If the target photomask is a back-end-of-line (BEOL) layer, the main drawn layer(s) used to produce this photomask (corresponding to the design’s intent or providing circuit completion) should be considered a layout base layer(s). Other layers above contact should NOT be base layers.
- Any layer not used to produce a photomask should NOT be designated as a base layer.
- If multiple base layers include data that has physical proximity, but deep inconsistent branches of hierarchy, then the base layers should be limited to represent only the more efficient branch. For example, fill layers with inconsistent hierarchy from the primary patterns generally should NOT be included in base layers.
- The optimum base layer configuration depends somewhat on design style, and may require some characterization for a production flow. Choice of the configuration should take into account the variety of design styles that may occur in the input layouts.

  The following is an example of an effective base layer strategy for a general CMOS process:

Target Photomask	Layout Base Layers
contact or below	active, poly, contact
metal1	metal1
metalN	metalN
viaN	viaN

Related Topics

LAYOUT BASE LAYER [Standard Verification Rule Format (SVRF) Manual]

LAYOUT TOP LAYER [Standard Verification Rule Format (SVRF) Manual]

Direct Data Access

The traditional Calibre platform supports three ways of handling data.

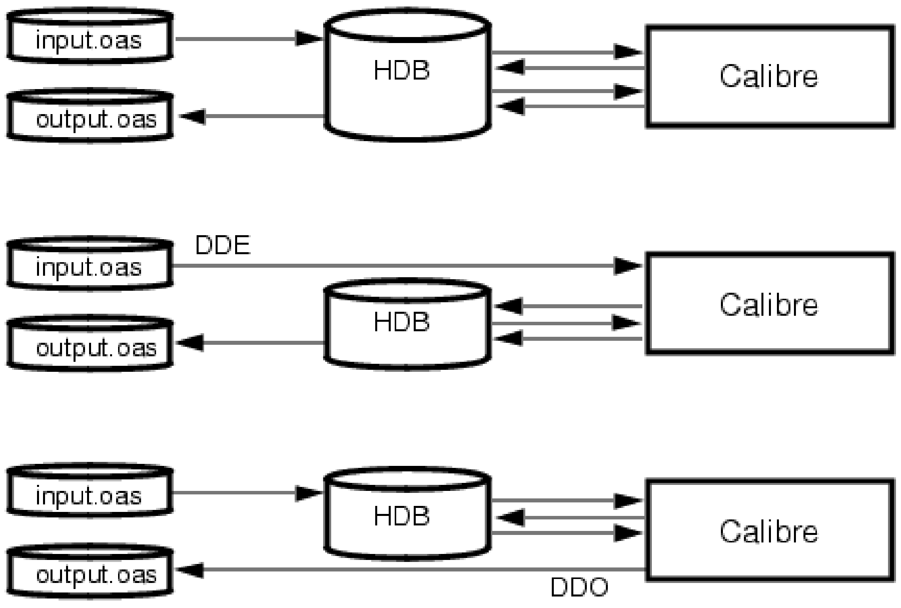
- **HDB** — The design data is stored as a hierarchical database in memory. Typically this resides on the primary host, but can also use remote memory.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- **Direct Data Entry (DDE)/Direct Data Output (DDO)** — The design data is directly read from (DDE) the input file and written to (DDO) the output file. This reduces the runtime by eliminating the HDB construction step.
- **DDE/DDO in combination with HDB** — The design data can be split between DDE, DDO, and HDB on a per-layer basis, including both dual input and dual output. No layers may go to both the HDB and input/output file, however.

The HDB mode is better for layouts that still have good hierarchy. DDE and DDO are most often useful for very flat design databases such as occur in EUV OPC correction or MDP. They should never be used with typical physical verification runs because they disregard layout hierarchy.

Figure 3-1. HDB versus DDE/DDO



Note

Prevent file conflicts by using a link in the local run directory to the input OASIS file instead of a direct reference. This is especially important when multiple jobs refer to the same layout. When you use a local link, index files are written locally rather than to the shared directory.

The Calibre FullScale platform uses a different data path than traditional hierarchical Calibre. For Calibre FullScale rule files, you do not need to specify LAYOUT BASE LAYER, TURBOflex, or ULTRAflex. You also do not specify direct_input, direct_output, or vboasis_path. All Calibre FullScale runs create index files.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

DDE Limitations

DDE has specific requirements:

- DDE requires litho flat mode.
- Only VB:OASIS format files are supported.
- The Calibre engine still requires an HDB, though it can be empty. The HDB is set by the LAYOUT PATH statement.
- Layers handled with DDE cannot be manipulated by SVRF statements before the Litho operation; that is, all DDE layers must be drawn layers, not derived.

  Only the following operations support DDE:

LITHO DENSEOPC	LITHO MASKOPT	LITHO MPC
LITHO NMBIAS	LITHO OPCVERIFY	RET PXOPC

DDO Limitations

The DDO flow has the following limitations:

- It is only supported in litho flat mode.
- You must set the CALIBRE_MTFLEX_LOCAL_HOST_DIR environment variable and the LOCAL HOST DIR configuration statement, as described in the *Calibre Administrator's Guide*.
- DDO data is flat, which produces large files.
- Only polygon output is supported. Any edge results or paths must be expanded. Annotations and text are not supported.
- ASCII output must be sent to an HDB. The HDB is set by the DRC RESULTS DATABASE statement.
- Any layers output via DDO cannot be manipulated by SVRF operations after the call.

  Only the following operations support DDO:

LITHO DENSEOPC	LITHO MASKOPT	LITHO MPC
LITHO NMBIAS	LITHO OPCVERIFY	RET PXOPC

Related Topics

direct_input

direct_output

Data Paths for Calibre FullScale

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Calibre Hierarchical Platform

The traditional Calibre platform uses the Calibre core hierarchical engine, the same processing engine as is used by physical verification. It offers some features designed for post-tapeout conditions.

When you start Calibre with a command beginning "calibre -drc -hier" the run is using the traditional Calibre platform.

Overview of Calibre Post-Tapeout Options

Calibre offers several additional features designed for post-tapeout input, which each handle hierarchical layout data differently.

Construction Methods

In order of introduction, these are the modes that use the Calibre hierarchical engine, the traditional platform used for pre-tapeout runs:

- **TURBOflex** — A data construction method that can be employed with standard hierarchical processing modes. TURBOflex changes some input hierarchy but not all. This makes it generally the best choice for RET/OPC flows, which tend to cause significant hierarchical degradation.
- **ULTRAflex** — A data construction method that can be employed with standard hierarchical processing modes. ULTRAflex changes almost all of the input hierarchy. It is recommended for runs that have inefficient or degraded hierarchy in the input.
- **MONDOflex** — A data construction method that can be employed with standard hierarchical processing modes. MONDOflex flattens all very small cell placements and "smallish" top-layer cell placements. *It is not suitable for pre-tapeout flows.* It should not be used with connectivity. TVF should be limited to conditional controls for determining sections of SVRF to execute at runtime.

The standard hierarchical methods are described in "Multithreaded Processing Modes" in the *Calibre Administrator's Guide*. The post-tapeout construction methods are described in "Which Construction Method to Use" on page 29.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Processing Modes

The processing modes determine how much hierarchy within a cell is maintained. In order of introduction, they are as follows:

- **Litho Hierarchical** — Also sometimes referred to as “standard hierarchical”. Input hierarchy is preserved and tiling is done on a per-cell basis.
- **Litho Flat** — Ignores hierarchy. Instead, it creates tiles across the entire extent of the input.
- **Litho Hybrid** — Uses an algorithm to determine which cell hierarchy is worth preserving, and flattens the rest.

For more details on processing modes, see “Which Processing Mode to Use” on page 34.

Data Paths

When data is passed between post-tapeout operations, it can be written in a central file, written to a distributed file and centralized once the run is complete, or streamed between operations without writing intermediate files.

- **Hierarchical Database (HDB)** — This is the traditional Calibre method, which makes use of a single file containing all results. It is described in “Hierarchical Database Construction” in the *Calibre Verification User’s Manual*.
- **Remote Data Servers** — Remote data servers (RDS) are also used in pre-tapeout runs. The RDS provides a distributed memory, which can improve throughput on a network.
- **Direct Data Access** — The direct data methods can be combined with HDBs but can also directly read and write data from the input and output.

For more details on data paths, see “Direct Data Access” on page 24.

Related Topics

Which Construction Method to Use

Which Processing Mode to Use

Calibre FullScale Platform

Multithreaded Processing Modes [Calibre Administrator's Guide]

Data Processing Modes [Calibre Administrator's Guide]

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Which Construction Method to Use

In most Calibre applications, the default standard hierarchical processing is the most efficient method due to the inherent database hierarchy that is present in most designs. The exceptions are certain jobs that have little or no incoming hierarchy (for example, post-Litho operations) and Calibre jobs that cause significant hierarchical degradation during a run (for example, Litho or fill operations). These cases generally contain pre- or post-Litho SVRF operations. The use of standard hierarchical processing in such cases can lead to severe performance degradation.

For only these cases, consider TURBOflex or ULTRAflex to improve the scalability and reduce the overall runtime. For cases that have good scaling, TURBOflex and ULTRAflex may lead to longer runtime due to overhead associated with the processing method.

TURBOflex is recommended over ULTRAflex for most post-tapeout flows because it retains more of the input database hierarchy, which can reduce the total computational burden. Exceptions to this recommendation include, but are not limited to, post-tapeout flows that are known to have low hierarchical efficiency in the input databases.

When you are unsure which is a better solution, investigate TURBOflex performance first.

The use of either TURBOflex or ULTRAflex in a traditional DRC flow with efficient hierarchy is strongly discouraged.

Table 3-1 is a summary of the general recommendations. This should be used as a starting point to chose a construction method. Optimization for a production environment requires some characterization of a sample of the population of layouts with the specific rule file. The best method for production may not be the fastest for every individual case.

Table 3-1. Recommended Processing Mode by Situation

Condition	Recommended Method
Known efficient input hierarchy Generally in the DRC flow	Default data construction (Calibre MTflex or MT)
Marginal or unknown hierarchical efficiency Generally in the PTO flow Jobs with pre- or post-Litho operations	TURBOflex
Flat input Very inefficient or degraded hierarchy	ULTRAflex
Flat input Very inefficient or degraded hierarchy High scalability Bounded range operations	Calibre FullScale Platform

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Default Data Construction

Calibre has two multi-processing modes which are used to distribute the large amount of computation required for post-tapeout flows across multiple CPUs.

The two processing modes are

- **Calibre MTflex** — For multiple networked remote hosts; ideal for the very large numbers of processors commonly used in the post-tapeout flow.
- **MT** — For a single primary host with multiple CPUs.

Hyperscaling is an additional mode that increases the efficiency of multi-processing by enabling the concurrent, parallel execution of SVRF operations. It is available for both Calibre MTflex and MT modes.

Standard hierarchical mode reads in the layer, and then constructs an internal representation that reflects the logical cell hierarchy. Each cell can be processed by a different thread. However, parent cells cannot be processed until child cells complete. Also, any time a cell instance is different from other cell instances, as is common post-OPC, the instance must be treated separately. This shows as degraded hierarchy. The combination of degraded hierarchy and hierarchical dependency are why standard hierarchical mode is a poor choice for post-tapeout flows.

Related Topics

Running MT and Calibre MTflex with Hyperscaling [Calibre Administrator's Guide]

TURBOflex Data Construction

Calibre TURBOflex is a construction method that can achieve highly scalable parallel processing on large, multi-CPU servers and distributed compute systems (clusters). TURBOflex is most useful in a post-tapeout flow that significantly modifies the database hierarchy during processing, and should be used with extreme caution in traditional DRC flows on highly hierarchical databases.

TURBOflex changes parts of the input hierarchy to produce a uniform and hierarchical structure of bins. This structure can be processed with high scalability on large, multi-CPU servers and distribute compute systems (clusters). This added hierarchy consists of seven tiered levels with a 64x64 grid of cells at the lowest level. The size of the cells is determined by the data extent of the input database. This data construction approach introduces potential performance improvements through scalability at the expense of a loss in hierarchical efficiency.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

After data construction and during processing, data may be promoted into the tiered cell structure above before final processing is completed. The promoted data will be used by subsequent SVRF operations and ultimately reflected in the output database.

During data construction, TURBOflex may place input data or cells at any level of the tiered hierarchy. Input data or cells may be present at any level of the TURBOflex hierarchy bins with the possibility of data promotion present in the output database. Where possible, it maintains the names of some input cells, but a significant amount of the input cell structure will not be present in the output database.

TURBOflex does the following:

- Maintains some of the original database cell structure/hierarchy.
- Allows remaining original cells to be located at any of the tiered hierarchy levels after construction of the Calibre database.
- Maintains the original primary cell name.
- Typically runs faster compared to ULTRAflex.
- Automatically enables the PSEUDO option to DRC Results Database.

You should use TURBOflex:

- In post-tapeout flows (be sure to avoid traditional DRC-style usage).
- When hierarchy related issues are a suspected cause of long runtime.
- When more predictable runtime across multiple random post-tapeout flow jobs is more desirable than optimized runtime for each job.

The requirements for running TURBOflex include:

- DRC Results Database statements should output GDS or OASIS format data.

The recommendations for running TURBOflex include:

- A Calibre job that is run using the -turbo keyword and either -remote or -remotefile options.

  While it is possible to run TURBOflex in MT or single-threaded mode, it is generally not feasible in a production environment due to performance considerations.

TURBOflex is enabled in SVRF by use of the LAYOUT TURBO FLEX specification statement, or by the environment variable CALIBRE_TURBO_FLEX.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

ULTRAflex Data Construction

Calibre ULTRAflex is a construction method that can achieve highly scalable parallel processing on large, multi-CPU servers and distributed compute systems (clusters). ULTRAflex is most useful in a post-tapeout flow and should be used with extreme caution in traditional DRC flows on highly hierarchical databases. The overall runtime for ULTRAflex is typically, but not always, slower than TURBOflex.

ULTRAflex transforms incoming hierarchy very much like TURBOflex. It produces a uniform hierarchical structure of bins. This added hierarchy consists of 7 tiered levels with a 64x64 grid of cells at the lowest level. The size of the cells is determined by the data extent of the input database.

ULTRAflex is different from TURBOflex in that during data construction, ULTRAflex places input data and some input cells in the lowest level of the tiered hierarchy *only*. The remaining tiered levels contain no input data and will only be used during processing where data promotion occurs. This method preserves less of the input hierarchy than TURBOflex does.

After data construction and during processing for either TURBOflex or ULTRAflex, data may be promoted into the tiered cell structure above before final processing is completed. The promoted data will be used by subsequent SVRF operations and ultimately reflected in the output database.

ULTRAflex does the following:

- Does not keep any of the original database cell structure/hierarchy, except for some limited cells within the lowest tiered hierarchy level.
- Does not keep the original primary cell name.
- Typically faster at constructing the Calibre database.

The requirements for running ULTRAflex include:

- DRC Results Database statements should output GDS or OASIS format data.

The recommendations for running ULTRAflex include:

- A Calibre job that is run using the -turbo keyword and either -remote or -remotefile options.

  While it is possible to run ULTRAflex in MT or single-threaded mode, it is generally not feasible in a production environment due to performance considerations.

ULTRAflex is enabled in SVRF by use of the LAYOUT ULTRA FLEX specification statement, or by the environment variable CALIBRE_ULTRA_FLEX.

Related Topics

TURBOflex Data Construction

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

MONDOflex Data Construction

Calibre MONDOflex is similar to Calibre ULTRAflex but has a deeper hierarchy — it consists of 8 tiered levels, with 128x128 grid of cells at the lowest level. Use it for the very largest designs such as multi-chip masks.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Which Processing Mode to Use

Calibre has three processing modes (litho hierarchical, litho flat, and litho hybrid) ideal for post-tapeout operations. The modes are set with the processing_mode command in the litho setup file. The litho flat and litho hybrid modes are not recommended for layout databases with high redundancy.

The PTO-specific methods optimize performance by how they handle operations rather than leveraging hierarchy.

Not all SVRF operations or setlayer commands are supported for these modes.

All post-tapeout operations in a rule file should use the same processing mode for best results.

Table 3-2. Recommended Processing Mode by Situation

Condition	Recommended Mode
Contains some cell hierarchy Has sparse OPC operations, RET NMDP, or RET REFINE	Litho hierarchical
Full chip or smaller design with poor hierarchy Long-range or unbounded operations Has LITHO EUV, LITHO MASKOPT, RET DPSTITCH, or RET PXOPC operations	Litho flat
Design with unknown hierarchy Does not contain FRACTURE, LITHO EUV, LITHO MASKOPT, MDP, RET DPSTITCH, or RET PXOPC operations	Litho hybrid

Litho Hierarchical Processing Mode

The litho hierarchical mode is the processing mode used by default, and can be used by all post-tapeout operations. Its basic unit of processing is the tile. It is an efficient mode when the layout is extremely hierarchical.

Litho hierarchical mode is used under the following conditions:

- The litho setup file does not include a processing_mode command.
- The litho setup file specifies “processing_mode hierarchical”.
- The litho setup file specifies “processing_mode hybrid” and some cells contain usable hierarchy.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

In litho hierarchical mode, cell hierarchy is preserved according to which data construction method is used. Within cells, the data is divided into tiles to improve processing speed. (The size of a tile is set by the tilemicrons command.) If a cell is smaller than the specified tile size, it is processed as one tile. Tiles follow cell boundaries and may be any shape.

Some litho operations further divide tiles into frames. Frames are much smaller than tiles and always square.

This is the only processing mode available to the older post-tapeout operations:

Table 3-3. Operations That Must Use Litho Hierarchical Processing Mode

LITHO OPC	LITHO ORC	OPCBIAS V1
LITHO PRINTIMAGE	OPCLINEEND	RET FLARE_CONVOLVE
RET NMDP	RET NMDPC	RET REFINE

Related Topics

Recommended Settings for tilemicrons per Node [Calibre nmOPC User's and Reference Manual]

Litho Flat Processing Mode

The litho flat processing mode can be more efficient when you know in advance that a layout has degraded hierarchy, such as post-OPC, or you want to treat the polygon placements as flat.

Litho flat processing occurs when either RET GLOBAL LITHOFLAT YES occurs in the SVRF file or the setup file for the LITHO, RET, or OPC operation contains the “processing_mode flat” command or “OPTION PROCESSING_MODE FLAT” keyword.

Litho flat processing can be used independently of Calibre Distributed Flat, but the deprecated Calibre Distributed Flat platform requires litho flat. When Calibre Distributed Flat is invoked, litho flat is automatically used even if the setup file does not specify it. The run exits with an error if any post-tapeout operation sets the processing mode explicitly to a value other than flat.

Litho flat requires ULTRAflex. In this mode, the full chip is split into tiles, which are then processed separately. As of version 2014.2, litho flat also automatically enables ULTRAflex.

Litho flat processing is supported by the following LITHO operations:

Table 3-4. Operations That Support Litho Flat

LITHO DENSEOPC	LITHO EUV DENSEOPC	LITHO EUV OPCVERIFY
LITHO MASKOPT	LITHO MPC	LITHO NMBIAS
LITHO OPCVERIFY	OPCBIAS (V2 only)	OPCSBAR (V2 only)
RET DPSTITCH	RET MBSRAF	RET PXOPC
RET SBAR		

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Litho flat processing might also be used on large chips when the setup file contains “processing_mode hybrid”.

Related Topics

Litho Hybrid Processing Mode

Litho Hybrid Processing Mode

The litho hybrid mode has the best average performance, as it automatically selects whether to process areas of the input in litho hierarchical or litho flat mode.

To use litho hybrid mode, the litho setup file for supported operations (see Table 3-5) should contain the command “processing_mode hybrid.”

In litho hybrid mode, cells that are deemed “good” and all their subcells are processed hierarchically. Cells with poor hierarchy are processed using litho flat. This is not as efficient as litho flat for very flat layouts, or as litho hierarchical for clean hierarchies, but overall has the best average performance.

Litho hybrid mode is only supported for the following operations:

Table 3-5. Operations That Support Litho Hybrid

LITHO DENSEOPC	LITHO MPC	LITHO NMBIAS
LITHO OPCVERIFY	OPCBIAS (V2 only)	OPCSBAR (V2 only)
RET MBSRAF	RET SBAR	

Related Topics

Litho Flat Processing Mode

Litho Hierarchical Processing Mode

processing_mode

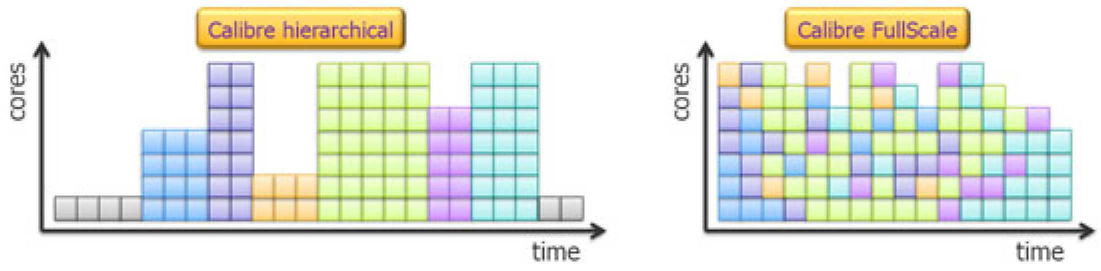
Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Calibre FullScale Platform

The Calibre® FullScale™ license unlocks a multi-processing platform designed for post-tapeout runs, which typically use a thousand or more CPUs and a flow of DRC, OPC, RET, and MDP products.

Calibre FullScale is a parallelization method that analyzes input for both independently executable sequences of SVRF operations and also independently constructed sections of layout. Independent operations are executed in parallel, and dependent operations are grouped and executed on a remote without saving intermediate results. Although it flattens the layout, it preserves significant hierarchy for large repeating cells such as multi-core CPUs.

Figure 3-2. Section-Based, Task-Based Processing



Part of the performance improvement comes from using direct data access instead of a hierarchical database. This imposes certain restrictions:

- The layout must be solely in GDS, OASIS, VSB, or MEBES jobdeck format. For example, it cannot be OASIS on top with GDS blocks.
- Only GDS and OASIS layouts may be compressed. VSB and MEBES cannot be in *.gz* format.

Because of the parallel operations and the data flattening, the remote CPUs generally require more memory than the Calibre hierarchical engine. Some of the memory load can be alleviated by using remote data servers, but 8 GB per CPU is still recommended. Network speeds of 10 GB/sec or better are also recommended to avoid having runs slowed by congestion.

Calibre FullScale is invoked with the -pto argument, used instead of -drc -hier. The typical invocation is

```
calibre -pto -turbo -remotefile distributed.cfg -remotedata rules.svrf
```

where *distributed.cfg* is the remote configuration file and *rules.svrf* is the post-tapeout rule file.

Some SVRF statements and setlayer commands have restrictions in Calibre FullScale runs, as detailed in the support sections.

Given the same input, the output from a Calibre FullScale run will be within 2 dbus to the output from a Calibre hierarchical engine run, except when the input has skew edges or an

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

intermediate operation produces skew edges. Skew edges snap to the database grid when clipped by a cell or section boundary, and the locations of the boundaries can be different.

License Consumption with Calibre FullScale

Calibre FullScale consumes licenses differently depending on whether it is using only Calibre MTflex (the default) or Calibre® Dynamic Resource Allocator (DRA).

Following are several examples. In all the examples, the number of licenses consumed is based on the standard multithreaded license consumption formula described in “Licensing for Multithreaded Operations” in the *Calibre Administrator’s Guide*. While some examples specify a value for -turbo, generally performance is better if no value is specified. The examples also use a cluster of 1000 CPU cores. The dotted lines in the figures indicate maximum number of CPU cores available.

Calibre MTflex Without a -turbo Value

The invocation is

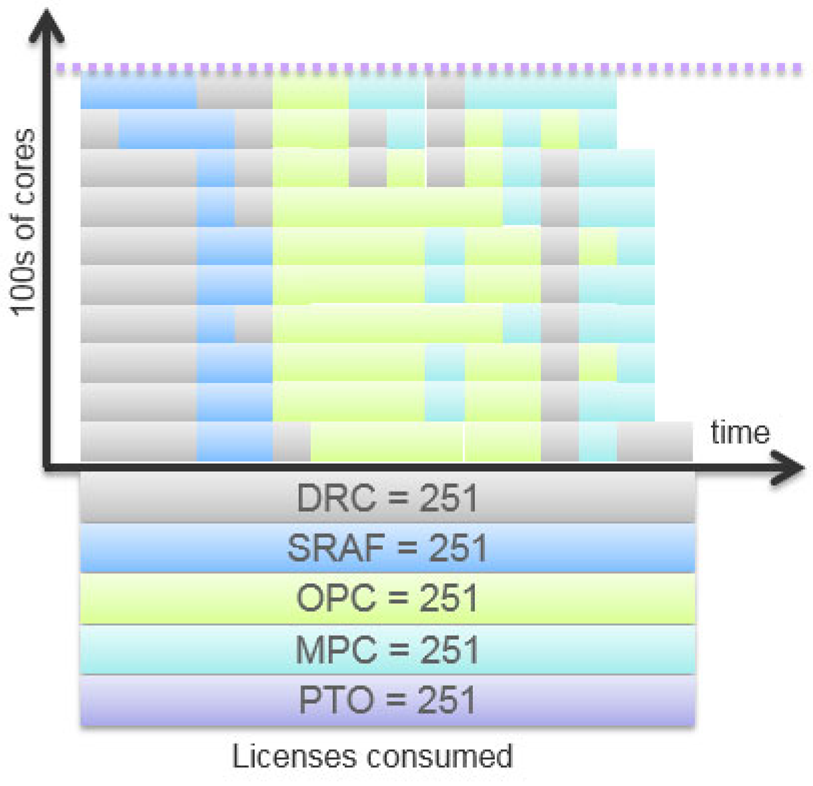
```
calibre -pto -turbo -turbo_litho example.svrf
```

All operations are licensed based on the total number of allocated CPU cores, and all licenses are consumed for the entire run.

For example, if the run uses the full cluster (1000 CPU cores), each licensed operation set (for example, OPCSBAR or LITHO MPC) uses 251 licenses as shown in the following figure.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 3-3. Licensing for Calibre MTflex Without a -turbo Value



Calibre MTflex With a -turbo Value

The invocation is

```
calibre -drc -turbo <N> -turbo_litho example.svrf
```

where <*N*> is less than or equal to the cluster size.

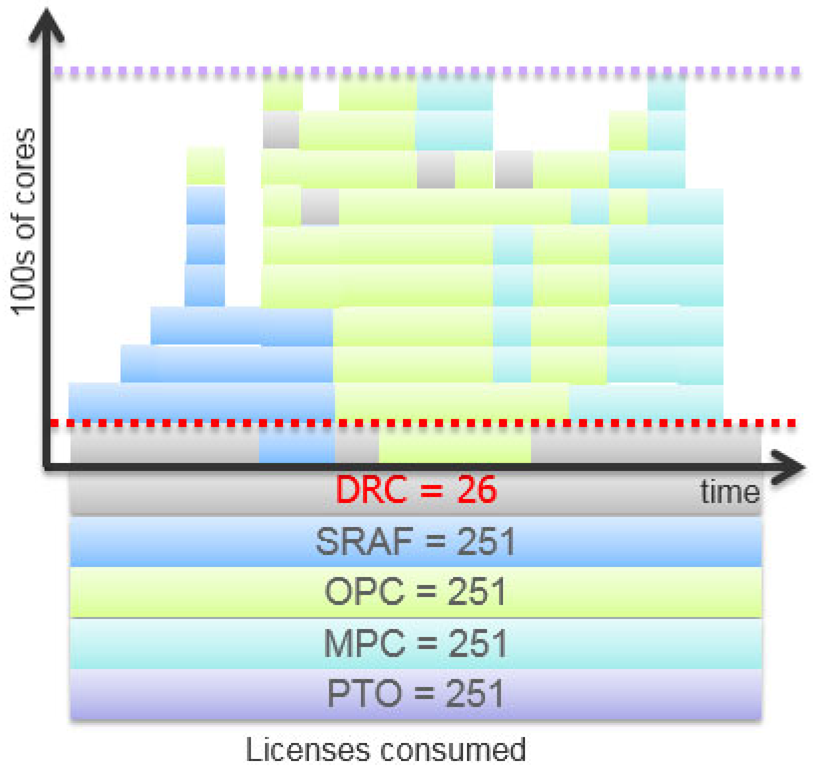
Post-tapeout operations are licensed based on the total number of allocated CPU cores and for <*N*> CPU cores for DRC operations. All licenses are still consumed for the entire run.

For example, if a value of “-turbo 100” is specified but no value is given for -turbo_litho, the run uses 26 DRC licenses. The post-tapeout operations use the full cluster and require 251 licenses as before, as shown in the following figure.

Notice that scalability is not as good as a similar run that does not restrict how many CPU cores can run DRC operations. Most post-tapeout runs perform DRC operations to create layers for the LITHO, RET, and MDP operations, and to write intermediate results or further modify output. With a value set on -turbo, some CPU cores may be idle while waiting for DRC operations to complete.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 3-4. Licensing for Calibre MTflex With a -turbo Value



Calibre DRA Without a -turbo Value

The invocation is

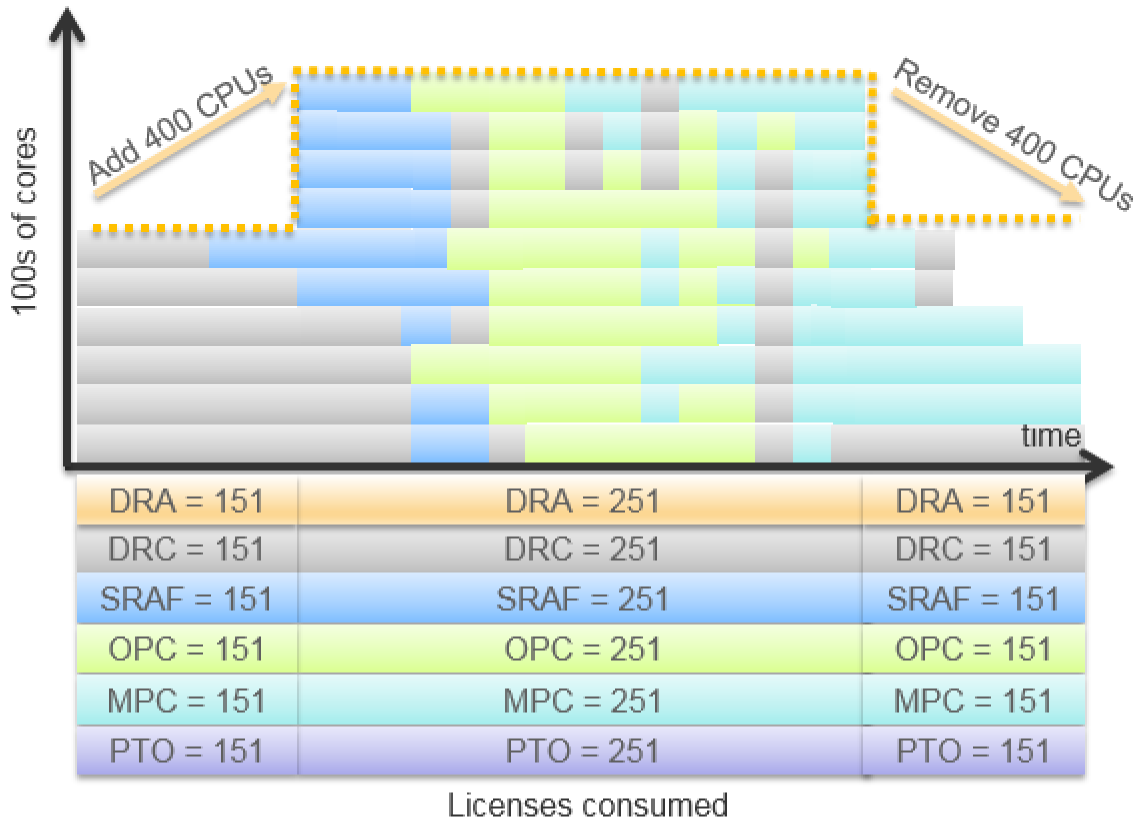
```
calibre -pto -turbo -dra example.svrf
```

This functions similarly to "Calibre MTflex Without a -turbo Value" on page 38 except that the number of allocated CPU cores (and thus the number of licenses consumed) can be changed during the run by Calibre DRA. (This is referred to as a DRA event.) Calibre DRA licenses are also consumed.

For example, a Calibre DRA with Calibre FullScale run may start with 600 CPU cores, consuming 151 licenses for all operations. The operator decides to add 400 CPU cores, which increases the licenses required to 251. After some time, the operator removes the 400 CPU cores again. This is shown in the following figure.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 3-5. Licensing for Calibre DRA



Calibre DRA With a -turbo Value

The invocation is

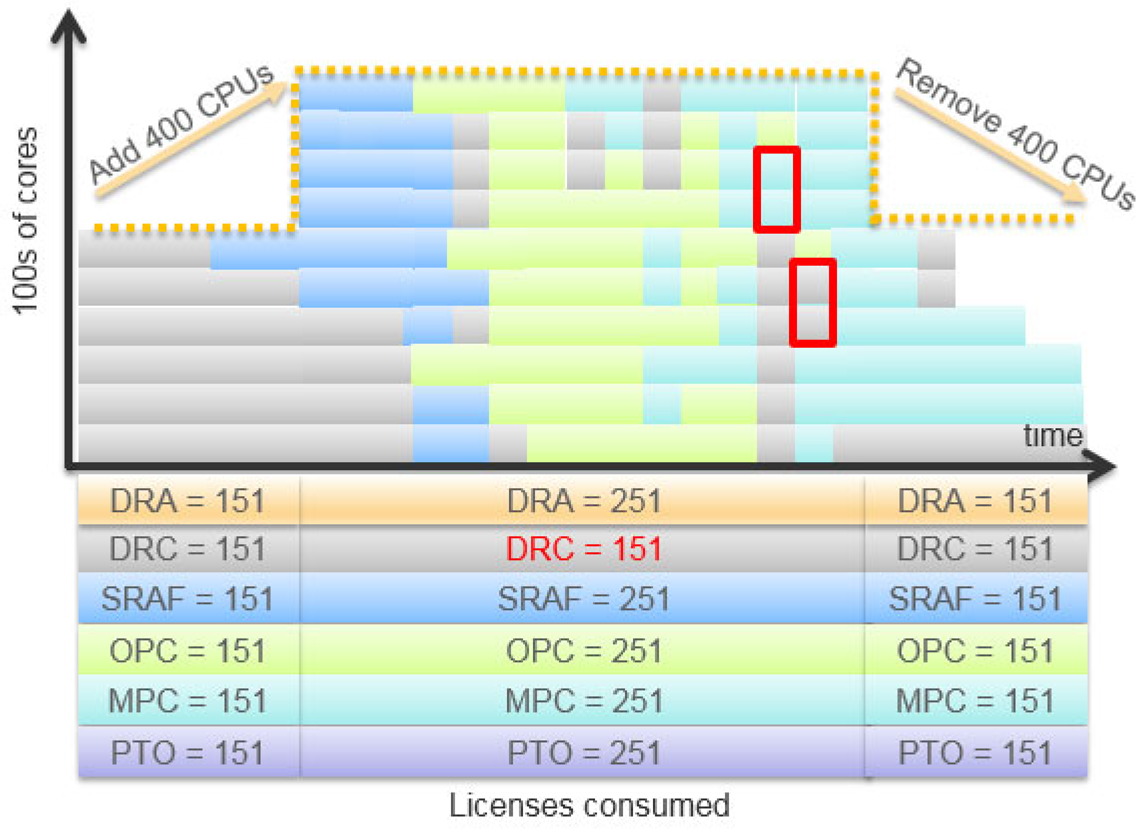
```
calibre -pto -turbo <N> -dra example.svrf
```

where <*N*> is less than or equal to the cluster size. In this case, the licensing for DRC operations is based on the number of CPU cores specified by <*N*>, while the post-tapeout operations can use up to the currently allocated cores.

For example, consider “-turbo 600”, with an initially allocated pool of 600 CPU cores. This begins similarly to the previous figure, but when 400 more CPU cores are added only the post-tapeout operations can use them. The changes are highlighted in red in the following figure.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 3-6. Licensing for Calibre DRA with a -turbo Value



In the example, the effect was that some DRC work happened later in the run as the number of licenses that could be checked out was limited by -turbo. Depending on the characteristics of the run, this could impact the overall time.

Related Topics

Calibre Dynamic Resource Allocator (DRA) User's Manual

Data Paths for Calibre FullScale

The thousands of remote hosts used by Calibre FullScale can stress network bandwidth. You can reduce some of the network congestion by carefully choosing how the intermediate data is stored.

Calibre FullScale efficiently loads the remote hosts by executing operations in parallel. By default in Calibre MTflex, the intermediate data is stored on the primary host RAM. The result is that the primary host can become a resource and performance bottleneck, which is why Calibre FullScale runs should use one of the methods below.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Two solutions to relieve network load at the primary host use different ways to distribute the intermediate layer data:

- Remote Data Server (RDS)
- Persistent FullScale Database (PFSDB)

If you are creating a new script to run Calibre FullScale and are not sure which to choose, use PFSDB. It provides a more scalable environment for Calibre FullScale runs provided that the configuration requirements are met.

Table 3-6. Recommended Data Path by Situation

Condition	Recommended Method
Not using Calibre FullScale	HDB, RDS, or DDE/DDO. See "Direct Data Access" on page 24.
Slow network filer access Remote hosts have sufficient RAM (8 GB/CPU + 2 GB per RDS process)	RDS
Unreliable remote hosts Performance constrained by remote-to-remote communication Using Calibre DRA	PFSDB

Remote Data Server (RDS)

With RDS, the intermediate data remains on the RAM of the remote hosts. For reliability, the invocation can also specify to create backups. This reduces the bandwidth needed by the primary host, and also its memory usage. The remote hosts providing the RDS service, however, require more memory than in the default case, and there may be increased remote-to-remote communication.

To use RDS, invoke Calibre FullScale with -remotedata. The hosts for the RDS are automatically chosen. Each RDS process serves as a database with a maximum of 1 GB capacity. You can also create a backup RDS process with -recoverremote, placing a total memory load on the RDS hosts of the memory needed to run Calibre FullScale plus an additional 2 GB.

See "Remote Data Server" in the *Calibre Administrator's Guide* for details on how servers are assigned.

Persistent FullScale Database (PFSDB)

PFSDB is a disk-based alternative to RDS. It stores the intermediate data in files on a network partition. The partition must be reachable by the primary host and all remote hosts. Much of the

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

network traffic is between the remote hosts and the filer, with no remote-to-remote communication.

The performance of PFSDB is determined by the network bandwidth between the network partition and the remotes and to the filer access speed. The PFSDB performance statistics are reported in the transcript with lines beginning with "<PFSDB:" immediately after the PROGRESS reports. The individual statistics are described in detail in "CALIBRE_FS_DB" on page 83.

PFSDB is set by an environment variable, CALIBRE_FS_DB. The Calibre FullScale invocation when this is set should look like

```
calibre -pto -turbo rules.svrf
```

It should not include the -remotedata argument.

Using MEBES and VSB Input

With some small changes, Calibre FullScale can read MDP rule files written for MEBES or VSB files and job decks.

SVRF files can supply an individual VSB or MEBES file or job deck in the LAYOUT PATH argument. The LAYOUT SYSTEM must be set to MFG. Any MDP EMBED rules must be moved to the main SVRF file.

Restrictions and Limitations

- VSB input must be VSB12 V2 or VSB12i.
- The input must not be compressed; that is, it cannot be in *.gz* or *.tar* format.

Procedure

1. Specify LAYOUT PATH and (optionally) LAYOUT PATH2.

   Only one filename can be provided. GOLDEN is not supported.

2. Set LAYOUT SYSTEM (and optionally LAYOUT SYSTEM2) to MFG.

   Before version 2020.2, Calibre FullScale required LAYOUT SYSTEM to be set to OASIS. As of version 2020.2, the OASIS setting when used with non-OASIS input causes the run to exit with an error.

3. Specify a single LAYER statement and number it 0. See the example.

4. Convert the MDP EMBED setup file.

   a. Copy the embedded SVRF (that is, the part between <SVRFSTART> and <SVRFEND>) to the main SVRF rule file.

   b. Remove any BOUND arguments from the relocated SVRF statements.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

   c. If the MDP EMBED setup file uses “jobdeck”, verify that LAYOUT PATH is using the same path.

   d. If the MDP EMBED setup file uses “svrf_layer_name”, convert the statement(s) to LAYER MAP statements.

5. Run the updated SVRF file with the -pto flag.

   ```
   calibre -pto -turbo mdp_input.svrf
   ```

   The MFG option for LAYOUT SYSTEM is only accepted when the rules are run with -pto rather than -drc.

Results

When the parser detects MEBES or VSB input, it reports <*CHIP/JOBDECK*> FILE NAME, <*CHIP/JOBDECK*> FILE PRECISION, and <*CHIP/JOBDECK*> FILE MAGNIFICATION in the transcript.

Examples

To specify a MEBES file:

```
LAYOUT PATH "/home/mdp_data/mebes.input"
LAYOUT SYSTEM MFG
LAYOUT PRIMARY "*"
LAYER mebes_data 0
```

The layer number 0 must be used to correctly map MEBES and VSB mask data.

Calibre FullScale SVRF Support

The Calibre FullScale platform is designed for post-tapeout needs. Some SVRF statements are not supported, or only partially supported. This section provides details on restrictions.

Note that some SVRF statements and litho setup file keywords have the same name, such as “size,” but could be supported differently. See “Calibre FullScale Setlayer Command Support” on page 51 for litho setup file keywords.

General Information

The SVRF language is divided into operations, which create layers, and specifications, which provide settings or write files but do not directly create layers. Operations that are not supported are reported in the transcript as “Forbidden Operations,” but specifications are not. For example, “DRC MAP TEXT DEPTH 1” is treated as “DRC MAP TEXT DEPTH ALL.” Operations that require awareness of geometry outside a section are supported, but Siemens support engineers may recommend rewriting such rules to improve run time.

For arguments to operations, generally, net-based filtering and connectivity arguments are not supported. The exceptions are CONNECT, INTERNAL, EXTERNAL, and ENCLOSURE.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

If a specification is not listed here with restrictions, it can be assumed to be supported.

SVRF Listing

Table 3-7. SVRF Operation Support for Calibre FullScale Runs

SVRF	Remarks
AND	
ANGLE	NOT ANGLE also supported.
AREA	NOT AREA also supported.
COINCIDENT EDGE	COINCIDENT INSIDE EDGE, COINCIDENT OUTSIDE EDGE, and “NOT” of all three forms are also supported.
CONNECT	CONNECT BY also supported.
CONVEX EDGE	
COPY	
CUT	NOT CUT also supported.
DEANGLE	
DENSITY	• Cannot use density expressions, WRAP, INSIDE OF EXTENT, GRADIENT, MAGNITUDE, PRINT ONLY, or RDB. • For INSIDE OF LAYER, cannot use CENTERED, BY EXTENT, BY POLYGON, or BY RECTANGLE. DENSITY CONVOLVE is *not* supported.
DFM CLASSIFY	• NOHIER is always in effect, even if not specified. • Cannot use ALL, KEEPPROPS, or DUPCOUNT.
DFM COPY	Cannot use UNMERGED, LAYERID, or CELL LIST.
DFM EXPAND EDGE	Cannot use UNMERGED.
DFM FILL	Only DFM FILL *spec_name* is supported. No options are supported.
DFM OR EDGE	

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 3-7. SVRF Operation Support for Calibre FullScale Runs (cont.)

SVRF	Remarks
DFM PROPERTY	• When you specify OVERLAP, the default behavior is MULTI, which is not the same as the hierarchical platform. • Cannot use NOMULTI, NOPUSH, UNMERGED, NODAL, CONNECTED, NOT CONNECTED, INSIDE OF, ACCUMULATE, BY NET, BY CELL, or DBU. • SPLIT PRIMARY is supported, but not SPLIT or SPLIT ALL. • The property definition cannot include PSPROPERTY, SVPROPERTY, NETPROPERTY, NETVPROPERTY, NETID, VNETID, or NARAC. • Even though the property definition can include vectors, the final result must be numeric or a numeric vector. DFM PROPERTY MERGE, DFM PROPERTY NET, and DFM PROPERTY SINGLETON are *not* supported.
DFM PROPERTY SELECT SECONDARY	• Cannot use NOMULTI, NOPUSH, or UNMERGED. • Cannot use :EXTENT or :EXTENT_EDGE. • Cannot use :EDGE on error cluster layers. • Measurements can only use original layers. Measurements on derived layers are not allowed.
DFM RDB	• Only syntax 1 is supported. • Cannot use more than one layer, NODAL, OUTPUT, JOINED, TRANSITION, ABUT ALSO, ANNOTATE, or CELL SPACE. • NOPSEUDO, SAME CELL and RESULT CELLS generate a warning that results may be different from results from the Calibre hierarchical engine. • ALL CELLS is not supported, but you can achieve a similar effect with CALIBRE_FS_RDB_DISABLE_INDEX.
DFM SEGMENT	
DFM SHIFT EDGE	Cannot use PRODUCED
DFM SPACE	Cannot use CONNECTED or NOT CONNECTED.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 3-7. SVRF Operation Support for Calibre FullScale Runs (cont.)

SVRF	Remarks

DFM SPEC FILL	• Only the following syntax is supported:
	`DFM SPEC FILL spec_name`
	`{{INSIDE OF x1 y1 x2 y2} |`
	` {INSIDE OF LAYER l_analyze}}`
	`[INITIAL FILLREGION [origin]]`
	`fill_pattern_definition`
	• Only one ***fill_pattern_definition*** is supported.
	• No optimizer or density analyzer is allowed.
	• For ***fill_pattern_definition***, RECTFILL and POLYFILL are supported, with the following options:
	STEP *step* | {*xstep ystep*}
	OFFSET *offset* | {*xoffset yoffset*}
	EFFORT *level*
	• For ***fill_pattern_definition***, the following STRIPE syntax is supported:
	`STRIPE width [separation]`
	`[HORIZONTAL | VERTICAL]`
	`[MINLENGTH min_len]`

DFM STRIPE	
DISCONNECT	
DONUT	NOT DONUT also supported.
DRC INCREMENTAL CONNECT	
DRC MAP TEXT	Hierarchy is not preserved; all text is written to the top cell. • DRC MAP TEXT LAYER is fully supported. • DRC MAP TEXT DEPTH is treated as ALL. • Text objects are included in the extent of the cell. This may change the database extent.
DRC RESULTS DATABASE	• Only OASIS and ASCII supported. • NOCBLOCK generates a warning; CBLOCK records are always generated. DRC RESULTS DATABASE PRECISION is also supported.
ENCLOSE	ENCLOSE RECTANGLE, NOT ENCLOSE, and NOT ENCLOSE RECTANGLE also supported.
ENCLOSURE	
EXPAND EDGE	
EXTENT	

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 3-7. SVRF Operation Support for Calibre FullScale Runs (cont.)

SVRF	Remarks
EXTENT CELL	Cannot use ORIGINAL, OCCUPIED, or WITH MATCH.
EXTENT DRAWN	Cannot use ORIGINAL, IGNORE, or layer specifications.
EXTENTS	Cannot use INSIDE OF LAYER.
EXTERNAL	
FLATTEN	
FRACTURE	• Only JEOL and vsb12 formats are supported. • Cannot use global options INSIDE OF EXTENT, INSIDE OF LAYER, or vboasis_path (use LAYOUT PATH instead). • Cannot use embedded SVRF blocks (<SVRFSTART> to <SVRFEND>).
GROW	
HOLES	
INSIDE	INSIDE EDGE, NOT INSIDE, and NOT INSIDE EDGE also supported.
INTERACT	NOT INTERACT also supported.
INTERNAL	
LAYOUT INPUT EXCEPTION SEVERITY	Supported exceptions are PRECISION_RULE_FILE, PRECISION_LAYOUT, and MISSING_REFERENCE.
LAYOUT PATH LAYOUT PATH2	Cannot use GOLDEN or STDIN. When multiple files are specified, all files must be OASIS and only appear once.
LAYOUT PROPERTY AUDIT	Cannot use APPEND.
LAYOUT SYSTEM LAYOUT SYSTEM2	Must be set to OASIS, GDS, or MFG.
LENGTH	NOT LENGTH also supported.
LITHO DENSEOPC	EUV supported. ML partially supported. Cannot collect training vectors (ML_VECTOR_CAPTURE_INIT and ML_VECTOR_CAPTURE_END).
LITHO MPC	
LITHO MPCVERIFY	
LITHO NMBIAS	
LITHO OPCVERIFY	EUV supported.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 3-7. SVRF Operation Support for Calibre FullScale Runs (cont.)

SVRF	Remarks
MAGNIFY	The magnification factor must be an integer.
MERGE	BY is ignored.
NOT	Commands that are supported are also supported with NOT keyword.
NOT WITH NEIGHBOR	Cannot use INSIDE OF LAYER.
OPCBIAS	V1 is automatically converted to V2.
OPCSBAR	
OR	OR EDGE also supported. ORNET is *not* supported.
OUTSIDE	OUTSIDE EDGE, NOT OUTSIDE, and NOT OUTSIDE EDGE also supported.
PERIMETER	
POLYGON	
PUSH	
RECTANGLE	NOT RECTANGLE, RECTANGLE ENCLOSURE, and RECTANGLES also supported.
RET SBAR	
SHIFT	
SHRINK	
SIZE	Cannot use OVERLAP ONLY or INSIDE OF LAYER.
SNAP	
TOUCH	TOUCH EDGE, TOUCH INSIDE EDGE, TOUCH OUTSIDE EDGE, and “NOT” of all four forms also supported.
VERTEX	
WITH EDGE	NOT WITH EDGE also supported.
WITH NEIGHBOR	Cannot use INSIDE OF LAYER.
WITH WIDTH	NOT WITH WIDTH also supported.
XOR	

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Calibre FullScale Setlayer Command Support

Many setlayer commands are available for use in litho setup files. Commands that are not supported that are in a litho setup file will be listed under "Forbidden Operations" in the transcript.

Generally, all setlayer commands listed in the Calibre OPCverify "Setlayer Operations Reference" are fully supported. Product-specific setlayer commands are supported if the product's SVRF operation is supported.

The following table lists the exceptions to setlayer command support. Setlayer commands that are fully supported do *not* appear in the table.

Table 3-8. Setlayer Commands With Limitations in PTO Mode

Command	Unsupported Functionality
dofcheck	common_ldof and sgd_output in add_properties
gate_stats	no_split_marker
gauges	• Input layer(s) of type "edge" • Properties copied from input layer • common_ldof and sgd_output in add_properties
image	VT5 models
interact	copy_props
pwcheck	common_ldof and sgd_output in add_properties
shadow_bias	VT5 model

Calibre FullScale Model Support

Calibre sometimes releases new types of models to handle the accuracy challenges of newly developed manufacturing processes.

The lists here pertain only to the Calibre FullScale platform. Some types of models may be supported by the platform but not with a particular product or feature. All models should be contained within a litho model.

- **Optical Models**
  - DUV optical models
  - DDM, including HHA, crosstalk, and corners.
  - EUV models, including black border, shadow bias and flare.
  - Topographical models (topo)
- **Resist Models**

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

    - CM1, including NTD

  VT5 resist models are not supported.

- **Process Models**
    - Variable etch bias (etch keyword)
    - 3D resist models (also called Zplanes)

  Compact DSA (CDSA) models are not supported.

- **MPC Models**
    - E-beam
    - Variable etch bias (vetchbias keyword)

  Long range models are not supported.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Chapter 4
Interpreting the Calibre Transcript for PTO Applications

This chapter is divided into two sections: An overview of the post-tapeout transcript, and links to descriptions of application-specific transcript information.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Transcripts for Post-Tapeout Operations

The post-tapeout operation transcript is inserted in the Executive Module of the overall Calibre run. As of release 2013.2, each LITHO operation that uses setlayer commands has a consecutively numbered transcript block.

Note

The sparse OPC operations do not generate these sections.

Sections are prefaced with a code to indicate location. The codes and the sections they occur in are listed in Table 4-1.

Table 4-1. Litho Transcript Sections and Prefixes

Prefix	Description
Lint Check sections	
Litho Header (Hierarchical Engine Only) section or FullScale Sections	
EV:	Environment variables.
SE:	Setlayer engine settings.
Tile Processing (Hierarchical Engine Only) section or FullScale Sections	
IF:	Initial Estimates for Tile Efficiency
Run Analysis section	
MS:	Overall Performance Viewed from Primary Host
TL:	Thread Performance for Tile Processing
PR:	Thread Performance Per mm^2 of Flat Design Area
FF:	Final Tile Efficiency Report
Detailed Run Analysis section	
DMS:	Overall Performance Viewed from Primary Host
DPP:	Primary Host Performance: Pre/Post-Processed in More Detail
DTL:	Thread Performance for Tile Processing
DSL:	Thread Performance for Setlayer Commands by Category
DMBS:	Performance for Model-Based SRAF Operations (mbSRAF only)
DPX:	Performance for pxOPC Operations (pxOPC only)
DII:	Performance for Dense OPC Operations (nmOPC only)
DIM:	Thread Performance for Imaging Operations
DPF:	Extra Processing Time for Litho Flat Mode

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 4-1. Litho Transcript Sections and Prefixes (cont.)

Prefix	Description
DPR:	Thread Performance Per mm^2 of Flat Design Area
MM:	Dense OPC Memory Tracker (nmOPC only)
MP:	Dense OPC Memory Tracker Percentiles (MB) (nmOPC only)
EC:	Tile-Level Raw Edge Counts For Contours
SSC:	SOCS FFT Performance

Lint Check

Some Calibre setup files are run through a “lint check” that validates settings for internal consistency. The check results appear in the STANDARD VERIFICATION RULE FILE COMPILATION MODULE section of the transcript.

The lint check sections appear in the transcript after the pathname of the rule file and before the contents. Each check begins with a header like the following:

```
// ===============================================================
// Lint Check for PXOPC Setup File pxopc.in (LITHO RUN 2)
// ===============================================================
```

This particular header indicates the setup file (*pxopc.in*) is used in a RET PXOPC operation, which is the second LITHO, RET, or MDP operation to run.

Lint checks can flag deprecated syntax, settings that are likely to hurt runtime, and violations of best practices. Correcting the items they warn of can improve both the quality of results and performance.

Related Topics

Rule File Compilation [Calibre Verification User's Manual]

lint [Calibre nmOPC User's and Reference Manual]

Global Calibre FRACTURE Arguments [Calibre Mask Data Preparation User's and Reference Manual]

Litho Header (Hierarchical Engine Only)

The Litho header section appears in the transcript after the Executive Module starts. It begins with the lines similar to the following:

```
=================================================================
=====        START DENSEOPC EXECUTION (LITHO RUN 1)         =====
=================================================================
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

The different operations provide different keywords for execution, but “LITHO RUN” is always the same.

The different operations also produce different subsections. Typical subsections are as follows:

- **Setup File** — Echoes the Litho setup file, including comments. It terminates with the following line:

  ```
  # --- END SETUP
  ```

- **Models** — Echoes the optical and resist or litho model information.
- **Environment Variables** — This begins by indicating which prefixes are checked for. It includes both MGC_HOME and the CALIBRE_ variables that are also in the DRC section of the transcript. Lines in this section are prefaced by “EV:”.
- **Setlayer Engine Settings** — Lists information such as process window conditions, optical and resist models, and the grid sizes. Lines in this section are prefaced by “SE:”.
- **Setlayer Execution Preview** — Indicates which operations are run concurrently. A concurrent group is one in which the operation runs once to produce multiple layers. The preview indicates in which order operations are run, and when layers are deleted relative to the operations.

Note that some operations, such as OPCSBAR, do not produce any subsections.

For a description of sections prior to the Litho header, see “Session Transcripts” in the *Calibre Verification User's Manual*; they are part of the DRC operations.

Tile Processing (Hierarchical Engine Only)

The tile processing sections provide enough information for run monitoring. The timing information is moved to timing tables in the detail section. Results are saved into separate summary tables.

The subsections include the following:

- **Init for Tile Processing** — Reports if litho-flat is used and time for preprocessing.
- **Initial Estimates for Tile Efficiency** — This section is prefaced “IF:”. It reports area-based statistics and calculates an efficiency factor. See “collect_frame_stats” in the *Calibre OPCverify User’s and Reference Manual* for details on the statistics.
- **Hier Mode Cell Processing** or **Flat Mode Cell Processing** — Depends on processing_mode value. In both cases, this section provides an initial run description

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

detailing the processing mode type and number of physical and remote cores. This is followed by progress lines like the following:

```
PROGRESS: 2% TIMESTAMP 15 LVHEAP = 15/15/15 SUBOP HI (LITHO-RUN 1)
PROGRESS: 6% TIMESTAMP 22 LVHEAP = 14/15/15 SUBOP HI (LITHO-RUN 1)
PROGRESS: 10% TIMESTAMP 27 LVHEAP = 14/15/15 SUBOP HI (LITHO-RUN 1)
...
PROGRESS: 89% TIMESTAMP 164 LVHEAP = 20/23/23 SUBOP HI (LITHO-RUN 1)
PROGRESS: 95% TIMESTAMP 167 LVHEAP = 20/23/23 SUBOP HI (LITHO-RUN 1)
PROGRESS: 100% TIMESTAMP 170 LVHEAP = 20/23/23 SUBOP HI (LITHO-RUN 1)
```

The section ends with time and memory reporting.

- **Postprocessing** — Echoes to the transcript at the start of each postprocessing operation, and reports the time as it completes.

Related Topics

Litho Flat Processing Mode

FullScale Sections

In Calibre FullScale runs, the information is split into two or more files, the Calibre FullScale log(s) and the standard output transcript.

Calibre FullScale logs are named *CalibreFSLog[.<host>].<process_id>[.<time>]*. The *<host>* and *<time>* sections of the name are used when the invocation includes -remote or -remotefile. Calibre FullScale logs contain detailed processing statistics and the run analysis tables.

The output transcript has information on the index file, section processing, operation construction, and summary statistics. The sections are as follows:

- **Setlayer Engine Settings**

  The setlayer engine settings that would normally be in the Litho header are written to a separate file named *CalibreFSLog.PreExec[.<host>].<process_id>[.<time>]* for Calibre FullScale.

- **Index File Information**

  Calibre FullScale requires an index file (*.fvi*) similar to that used in the Calibre FRACTURE tools. The index file information appears immediately after the Executive Module banner.

  Index files by default are searched for first in the current working directory and then in the directory containing the OASIS file. (Use a local link in the current working directory to the OASIS file to prevent multiple jobs causing a file conflict.) You can override the default using MDPIndexSearchPath, as described in "OASIS File

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Indexing” in the *Calibre MDPview User’s and Reference Manual*. If an index file is not found or it cannot be loaded, Calibre FullScale creates the index file.

- **Input Processing Information**

  Input processing includes layer filtering and cell injection.

  In the layer filter lines, the first number is the total number of objects found. The second number is the number of objects used after empty placements are filtered out.

  Cell injection consists of creating an internal hierarchy to query the database more efficiently. Duplicate layers are also aliased during this stage.

  Both Calibre FullScale and the Calibre hierarchical engine output the database extent. However, the two are calculated differently. Calibre FullScale uses the original OASIS file’s extent. The Calibre hierarchical engine uses the extent of the input layers required for the run.

  The final portion is the list of “OPs”, which are independently executable chunks of SVRF operations.

- **Task Construction**

  After analyzing the layout and operations, the Calibre FullScale engine constructs the tasks to process. Tasks are a unit of work, consisting of a section of layout and the OP to perform upon it. OPs are associated with multiple tasks.

  This section of the transcript begins with the line

  ```
  STARTING TASK GENERATION: LVHEAP =
  ```

  The number of tasks is listed at the end of the section a few lines after “FINISHED TASK GENERATION.” For example:

```
FINISHED TASK GENERATION: CPU TIME = 404  REAL TIME = 116 ; LVHEAP = 36024/36025/36025  RSS
= 43223  SHARED = 0/0  REMOTE = 1253/1280
Default layout grid: 68 x 55
STARTING EXECUTIVE PREPARATION: LVHEAP = 36023/36025/36025  SHARED = 0/0  REMOTE = 1253/1280
FINISHED EXECUTIVE PREPARATION: CPU TIME = 100  REAL TIME = 103 ; LVHEAP = 38973/38974/38974
RSS = 46059  SHARED = 0/0  REMOTE = 1253/1280
Generated 10719448 tasks; LVHEAP = 38973/38974/38974  SHARED = 0/0  REMOTE = 1253/1280
```

- **Runtime Information**

  A major portion of the Calibre FullScale transcript tracks execution. In addition to Calibre MTflex messages (MON: statements) there are OP blocks and progress lines.

```
PTO OP   44 (7567/10041/594/435)
    not_inside_layer.5 = line_end.4 TOUCH EDGE outside_edge_result.4
CPU TIME = 449  OP TIME = 31  REMOTE REAL TIME = 599  OPS COMPLETE = 66 OF 419  ELAPSED TIME
= 2401

PROGRESS: 10%; LVHEAP = 41217/41218/41218  SHARED = 0/0  REMOTE = 27114/28512  GEDB = 3284/
25795  ELAPSED TIME = 2401
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

The progress percentage refers only to the amount of completed tasks. It cannot be used to judge remaining runtime as some tasks take significantly longer than others.

If you are using PFSDB, it reports on filer performance immediately after the progress line. The PFSDB information is described in "CALIBRE_FS_DB" on page 83.

After all tasks are completed, the processing time and memory are printed:

```
PROCESSING TIME: CPU TIME = 1543 + 6109208  REAL TIME = 16785
                 REMOTE REAL TIME = 6411136
MEMORY: LVHEAP = 41216/41385/41402  SHARED = 0/0  REMOTE = 1801/30336.
```

To determine scalability, divide the remote real time by the real time and then by the remote count.

- **Scheduling Statistics**

  Scheduling statistics appear after the runtime information. The scheduling statistics section begins "SCHEDULING STATISTICS:" and is similar to the example below.

```
SCHEDULING STATISTICS:
TOTAL TASKS = 10719448  DISCARDED TASKS = 166184  PRUNED DEPENDENCIES = 924711
    REPORTING  TIME: CPU TIME = 4 + 0  REAL TIME = 5
    MEMORY: LVHEAP = 41216/41385/41402  SHARED = 0/0  REMOTE = 1801/30336.
MTFLEX OVERHEAD: AVERAGE 1.552 ms MAX 4583.973 ms
HISTOGRAM: [0.000000, 500.000000], step 50.000000
HISTOGRAM: 10543745    2350    348    208    408    79    15    4    5    1    35
AVERAGE MTFLEX DATA SIZE = 0.257207 KiB.
OVERHEAD TIME-SIZE CORRELATION COEFF: 0.0384658
UPDATE OVERHEAD: AVERAGE 1.318 ms MAX 4652.336 ms
WORM CACHE: 0 hits in 0 accesses.
    POSTOP TIME: CPU TIME = 0 + 44  REAL TIME = 53
    MEMORY: LVHEAP = 41021/41384/41402  SHARED = 0/0  REMOTE = 1801/30336.
BATCH HISTOGRAM: 166097    10545366
LAYERS PEAK / TOTAL MEMORY:
```

  Of particular note is the MTFLEX OVERHEAD line, which is correlated with network latency. The ideal values are a small average and a small maximum.

  The LAYERS PEAK / TOTAL MEMORY is followed by lines listing the memory usage for intermediate and final layers, and may help determine if some runs are causing memory overload. (The progress meter also reports intermediate layer memory use.)

There are some standard sections for Calibre hierarchical runs that do not appear in Calibre FullScale runs.

- Calibre FullScale does not use an HDB. Consequently, the results database initialization section does not appear and the following lines will not appear:

```
----- CALIBRE LAYOUT DATA INPUT MODULE -----
--- CALIBRE LAYOUT DATA INPUT MODULE COMPLETED. CPU TIME = 0 REAL TIME = 0
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

```
----- CALIBRE::DRC-H - RESULTS DATABASE INITIALIZATION MODULE -----
--- CALIBRE::DRC-H RESULTS DATABASE INITIALIZATION MODULE COMPLETED. CPU TIME = 0 REAL TIME
= 0
```

- Some run analysis performance tables are not produced, most notably the MS and TL tables.

Run Analysis

The RUN ANALYSIS section at the end of the transcript for each LITHO product run contains several tables that analyze the primary and remote performance for that run. Most of the tables are in standard formats.

The run analysis begins by echoing the processing mode and number of cores used. It includes several tables for evaluating performance. Generally the most useful tables are MS:, TL:, and FF:. (For more information on FF:, see "collect_frame_stats" in the *Calibre OPCverify User's and Reference Manual*.)

Each of these summary reports also has a detailed expansion in the RUN ANALYSIS: DETAILED section, except for Final Tile Efficiency. The table lines use a three-letter version of the prefix: D + summary code. For example, the detailed version of the MS: table is DMS:. The major items in the detailed tables are identical to those in the corresponding summary tables.

- "Performance Table Format" on page 60
- "MS: and DMS: Tables" on page 62
- "TL: and DTL: Tables" on page 62
- "DSL: Tables" on page 63
- "DIM: Tables" on page 63

Performance Table Format

Each line reports an operation. Some operation names are technical and of limited interest to end-users. The detailed table versions also report "Other". "Other" reports the difference between total time and the time assigned to specific timers. It should always be small.

Some lines are indented. Indented sections are the details that sum to the values reported in the unindented line above the section.

Primary Host Tables

The MS:, DMS:, and DPP: tables report on the statistics of the primary CPU. They are the only tables that contain elapsed "wall clock" time.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 4-2. Column Headings for MS, DMS, and DPP Tables

Column Heading	Meaning
operation	The name of the task being measured.
totcpu or mcpu(s)	MT runs report totcpu, which is the total CPU time for both the primary and all remote hosts. Calibre MTflex runs report mcpu(s), which is the CPU time in seconds for the primary CPU only.
mreal (s)	The elapsed time for the operation in seconds.
rcpu (s)	The total remote CPU time in seconds.
rc/mr	The ratio of rcpu to mreal. It is a measure of remote utilization. The best case is when it is close to the number of physical remote cores.
endtime	The elapsed time at the end of the operation in seconds. It is measured from the start of the Calibre execution.
count	The number of times a timer was activated. This number is usually not useful because timers can be started multiple times during one operation.

Remote Tables

The TL:, DIM:, PR:, and FF: tables report on the performance statistics of the remote CPUs.

Table 4-3. Column Headings for TL, IM, and DIM Tables

Column Heading	Meaning
operation	The name of the task being measured.
cpu (s)	The total remote CPU time in seconds, summed over all remotes.
real (s)	The total remote elapsed time in seconds, summed over all remotes. Note that because the remotes run at the same time, this is *not* the same as "wall clock" time.
count	The number of times a timer was activated. See the "TL: and DTL: Tables" section.
avgreal	The average real time for processing one tile.
maxreal	The longest real time measured for any tile.
tile	The index number of the tile in maxreal.
cell	The hierarchical cell containing the tile from maxreal.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

MS: and DMS: Tables

The MS: and DMS: tables show an overview of the run. Here is an example of an MS: table:

```
MS:-----------------------------------------------------------------------------
MS:| SUMMARY: Overall Performance Viewed from Master
MS:----------------------------------------------------------------------------
MS: operation                    mcpu(s)  mreal(s)   rcpu(s)  rc/mr  endtime  count
MS:-----------------------------------------------------------------------------
MS: All PREVIOUS Calibre Work 129.76     130.0       422   3.25      130       1
MS:
MS: Current Litho Product Run  118.51   10500.0    157958  15.04    10630       1
MS:
MS:   PreProcessing             22.62      45.4       317   6.98      176       1
MS:   Flat: Cell Processing     92.08   10452.9    157637  15.08    10629       1
MS:   PostProcessing             3.81       1.6         4   2.60    10630       1
MS:-----------------------------------------------------------------------------
```

The line All PREVIOUS Calibre Work shows all work in this Calibre run *before* the current LITHO run started.

For LITHO OPCVERIFY, the time for PostProcessing in MS: refers primarily to primary-level output_window and error classification. (These also have a tile-level component.) For other products, this item is usually negligible. The DMS: table provides more details.

TL: and DTL: Tables

The TL: and DTL: tables show remote performance for all tile processing. Here is an example of a TL: table:

```
TL:---------------------------------------------------------------------------------
TL:| SUMMARY: Performance for Tile Processing
TL:---------------------------------------------------------------------------------
TL: operation                        cpu(s)   real(s)  count  avgreal  maxreal  tile  cell
TL:---------------------------------------------------------------------------------
TL: All Tile Operations              157067    164876    756  218.090  296.840   413  TOP
TL:
TL:   Tile PreProcessing                124       127    711    0.178    0.365   716  ICV_3
TL:
TL:   Setlayer Operations            154847    162606    711  228.700  291.080   413  ICV_5
TL:     image (ctr_nom...)           135247    141749    711  199.366  242.579   413  ICV_5
TL:     veb_simulate (ctr_etch)       16215     17386    711   24.453   41.917   423  ARRAY
TL:     bridge (brdg)                  3365      3450    711    4.852    8.425   153  TOP
TL:
TL:   Tile PostProcessing              2096      2143    711    3.014    5.515   423  ICV_9
TL:---------------------------------------------------------------------------------
```

Lines under “Setlayer Operations” list the time for every individual concurrent group of setlayer operations. See the section “SETLAYER EXECUTION PREVIEW” for lists of concurrent groups. If a group contains more than one setlayer operation, only the name of the first output layer is shown, followed by ellipses (…).

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

For LITHO OPCVERIFY, the time for Tile PostProcessing in TL: refers primarily to tile-level output_window and error classification. For other products, this item is usually negligible. The DTL: table provides exact details.

The smallest value for count in the TL: table provides an accurate indicator of how many non-empty tiles were processed.

DSL: Tables

The DSL: tables show remote performance for setlayer operations aggregated by the operation name. (For example, all setlayer image operations are summarized on the line labeled DSL: image.) Here is an example of a DSL: table:

```
DSL:--------------------------------------------------------------
DSL: DETAILED: Master Performance for Setlayer Commands by Category
DSL:--------------------------------------------------------------
DSL: operation      cpu(s)  real(s)  count  avgreal  maxreal  tile  cell
DSL:--------------------------------------------------------------
DSL: image         21220.5  21427.5   3992     5.37    85.18   234  bhchipg1b
DSL: or            10872.2  10978.4   3992     2.75    55.84   307  bhchipg1b
DSL: and            6050.6   6112.0   3992     1.53    27.44   392  bhchipg1b
DSL: hard_failure   4643.4   4687.2  15968     0.29     7.57   298  bhchipg1b
DSL: copy             0.0      0.0   3992     0.00     0.00     0  ICV_23121$C1$
DSL: empty_layer      0.0      0.0   3992     0.00     0.00     0  ICV_27078$C0$
DSL:--------------------------------------------------------------
```

The DSL: table expands on the Setlayer Operations section of the TL: table. For LITHO OPCVERIFY, the pinching, bridging, and printing checks are grouped under hard_failure.

DIM: Tables

The DIM: tables show remote performance for image simulation. Only LITHO OPCVERIFY and LITHO DENSEOPC runs produce DIM: tables, because for these products image

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

simulation is the major component of performance. Here is a trimmed example of a DIM: table with minor items removed:

```
DIM:---------------------------------------------------------------------------
DIM:| DETAILED: Master Performance for Imaging Operations
DIM:---------------------------------------------------------------------------
DIM: operation                     cpu(s)   real(s)  count  avgreal  maxreal  tile  cell
DIM:---------------------------------------------------------------------------
DIM: Total Imaging Time             135247    141749    693  204.544  242.578   413   BBB
DIM:
DIM:   Initialize Models                 8         8    693    0.012    0.035   133  ICV_3
DIM:   Frame Striping                  131       134    693    0.193    0.415    60  ICV_2
DIM:   Rasterize Mask                 4329      4421  86880    0.051    0.189   156  ICV_3
DIM:   Mask RSM/DDM                  25201     25684  97740    0.263    0.817   217  ICV_3
DIM:   Make Mask Spectrum             7631      7795  86880    0.090    0.294    59  ICV_2
DIM:   SOCS Optics                   22032     22507  86880    0.259    0.671   219  ICV_3
DIM:   CM1 Resist Model              67623     72495  10860    6.675   10.971   127  ICV_3
DIM:   Make Frame Contours            3674      3748  32580    0.115    0.758   566  ICV_2
DIM:   Contour PostProcessing         1353      1397    693    2.016    3.911   423  ICV_2
DIM:---------------------------------------------------------------------------
```

Related Topics

collect_frame_stats [Calibre OPCverify User's and Reference Manual]

Product Manual Discussions

Post-tapeout users commonly want to analyze and optimize the performance of their Calibre jobs. Following are some references to assist with the interpretation of information contained in the Calibre transcripts.

General

- Overview of the contents of the transcript file: See “Session Transcripts” in the *Calibre Verification User's Manual*.
- Operation statistics (for example, FGC, HGC, LVHEAP): See “Executive Process” in the *Calibre Verification User’s Manual*.

Litho

- nmOPC Transcripts: See “Understanding the Transcript” in the *Calibre nmOPC User's and Reference Manual*.
- OPC Errors: See “OPC Operation Errors” in the *Standard Verification Rule Format (SVRF) Manual*.
- Geometry Counts: See “The LITHO OPCVERIFY Transcript” in “LITHO OPCVERIFY” in the *Calibre OPCverify User’s and Reference Manual*.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- Litho Errors: See “Litho Operation Errors” in the *Standard Verification Rule Format (SVRF) Manual*.
- OPCsbar Transcripts: See “Calibre Output Results” in the *Calibre OPCsbar User's and Reference Manual*.

MDP

- Transcripts: See “Transcript Messages” in the *Calibre Mask Data Preparation User’s and Reference Manual*.
- Exit Status: See “Status Values” in the *Calibre Mask Data Preparation User's and Reference Manual*.

Processing Modes

- Transcripts: See “Transcript Features Based Upon Run Mode” in the *Calibre Verification User's Manual*.
- Hyper Remote Transcripts: See “Hyper Remote” in the *Calibre Administrator's Guide*.
- Monitor Transcripts: See “Monitoring Remote Processes” in the *Calibre Administrator's Guide*.
- Calibre MTflex Errors and Warnings: See “Errors and Warnings” in the *Calibre Administrator's Guide*.
- Litho Run transcripts: See “Transcripts for Post-Tapeout Operations” in this chapter.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Chapter 5
Post-Tapeout Commands and Variables

Some data processing commands should be restricted to post-tapeout runs because of the significant impact they have on data hierarchy. To reduce the chance of misuse, the commands are not listed in the *Standard Verification Rule Format (SVRF) Manual*.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

calibre -pto

Initiates a Calibre run using the Calibre FullScale engine.

Usage

calibre -pto [-turbo [*number_of_CPUs*] [-turbo_all] [-turbo_litho [*number_of_CPUs*]]
[-remotefile *filename* -remotedata [-recoveroff | -recoverremote]]]
rule_file_name [-dra]

Arguments

- -turbo [*number_of_CPUs*] [-turbo_all]

  Specifies to use multithreaded parallel processing. The *number_of_CPUs* argument is optional; if not specified, Calibre runs on the maximum number of CPUs for which you have licenses.

  The -turbo_all option causes Calibre to exit with an error when it cannot secure enough licenses for the requested number of CPUs.

- -turbo_litho [*number_of_CPUs*]

  Specifies to use multithreaded parallel processing for LITHO, RET, and MDP operations. The *number_of_CPUs* argument is optional; if not specified, Calibre runs on the maximum number of CPUs for which you have licenses. The requested number of CPUs can be different from -turbo.

  The -turbo_litho option cannot be specified with the -dra option.

  Tip

  You do not need to specify both -turbo and -turbo_litho. If either one is specified, the entire run uses multithreaded parallel processing.

- -remotefile *filename*

  Specifies the path to a configuration file containing Calibre MTflex settings.

  For more details, see "Command Line Options Reference Dictionary" in the *Calibre Administrator's Guide*.

  Note

  You can use -remotecommand or -remote instead of -remotefile, though it is impractical to list the thousands of remote hosts typically used for a post-tapeout run.

  The -remotefile, -remotecommand, and -remote options can only be used with -turbo.

- -remotedata [-recoveroff | -recoverremote]

  These options control the behavior of the Calibre Remote Data Server. These options can only be used with -remote, -remotefile, or -remotecommand.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

These options are discussed under "-remotedata" in the *Calibre Administrator's Guide*.

For Calibre FullScale runs, -remotedata -recoverremote is recommended.

- ***rule_file_name***

  A required argument that specifies the pathname of the rule file.

  The Calibre FullScale architecture does not support all SVRF operations or all setlayer commands. See "Calibre FullScale Platform" on page 37 for details.

- -dra

  Specifies to use Calibre® Dynamic Resource Allocator (Calibre DRA), which permits changing the number of assigned CPUs during the run. This option cannot be specified with -turbo_litho.

Description

The -pto switch initiates a Calibre run using the Calibre FullScale engine, a separately licensed product. The parallelization features are designed for typical post-tapeout runs:

- Large design databases (full chips, or multi-chip masks) with degraded hierarchy
- Long chains of layer derivations
- Many hundreds or thousands of CPUs

Calibre FullScale is intended for use with taped out designs, and does not support all Calibre features. Layouts must be in GDS, OASIS, MEBES, or VSB format. For MEBES and VSB, compression (*.gz*) is not supported.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Post-Tapeout SVRF Commands

The post-tapeout SVRF commands should not be used in most physical verification runs.

The font conventions used in the command descriptions are described in "Syntax Conventions" on page 12.

Table 5-1. Commands for Post-Tapeout Flow Runs

Command	Description
LAYOUT CELL OVERSIZE AUTOLITHO	For hierarchical EUV flows. Sets Calibre to automatically calculate the best oversize value for the cell for the interactions in the rule file.
LAYOUT MONDO FLEX	Specifies MONDOflex processing mode.
LAYOUT RECOGNIZE DUPLICATE CELLS	Controls recognition of equivalent cells during the hierarchical database construction phase of hierarchical Calibre applications.
LAYOUT TURBO FLEX	Specifies TURBOflex processing mode.
LAYOUT ULTRA FLEX	Specifies ULTRAflex processing mode.
RET GLOBAL BOUND YES	Applies a global bound on all unbounded SVRF operations in Calibre Distributed Flat or MDP Embedded mode.
RET GLOBAL LITHOFLAT YES	Enables litho flat mode for all setlayer-based RET and LITHO commands.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

LAYOUT CELL OVERSIZE AUTOLITHO

SVRF statement

For hierarchical EUV flows. Sets Calibre to automatically calculate the best oversize value for the cell for the interactions in the rule file.

Usage

LAYOUT CELL OVERSIZE AUTOLITHO

Arguments

None.

Description

This optional command is intended primarily for use in the hierarchical EUV flow. It affects all cells in the design and changes the cells' extents prior to constructing the hierarchical database.

This statement may be specified at most once per rule file.

LAYOUT CELL OVERSIZE AUTOLITHO calculates an EUV-appropriate oversize of the cell extent in order to limit hierarchy degradation due to the long-range effects of flare.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

LAYOUT MONDO FLEX

SVRF statement

Specifies MONDOflex processing mode.

Usage

LAYOUT MONDO FLEX {**NO** | **YES**}

Arguments

- **NO** | **YES**

  Required keywords that indicate whether MONDOflex is to be used. Specify either NO or YES, but not both.

  NO represents the default behavior when the statement is not included in the rule file.

Description

An optional SVRF statement that specifies the rule file is run with MONDOflex. This statement may only appear once per rule file. If the statement appears additional times, the run exits with an error:

```
ERROR: Error SPC1 on line N of file - superfluous specification statement:
layout mondo flex.
```

This error message also appears if either LAYOUT TURBO FLEX and LAYOUT ULTRA FLEX are set to YES.

When MONDOflex mode is used, the transcript includes the line “MONDO FLEX ENABLED”. Some things may cause MONDOflex mode to not be used even if this statement is in the rule file:

- If the environment variables CALIBRE_ULTRA_FLEX or CALIBRE_TURBO_FLEX are set.
- If the Calibre run is invoked with the -flat option. In that case, the Calibre Distributed Flat platform, which uses ULTRAflex, is used.
- If the Calibre run is invoked flat (that is, it does not include the -hier or -flat option), MONDOflex is not used.

Examples

LAYOUT MONDO FLEX YES

Related Topics

Overview of Calibre Post-Tapeout Options

CALIBRE_MONDO_FLEX

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

LAYOUT RECOGNIZE DUPLICATE CELLS

SVRF statement

Controls recognition of equivalent cells during the hierarchical database construction phase of hierarchical Calibre applications.

Note

This statement is not recommended because it sometimes degrades performance and can report fewer DRC errors than actually exist.

Usage

LAYOUT RECOGNIZE DUPLICATE CELLS {**NO** | **YES**}

Arguments

- **NO**

  A required keyword that specifies recognition of equivalent cells is not performed. This is the default behavior.

- **YES**

  A required keyword that specifies recognition of equivalent cells is performed.

Description

This statement controls recognition (and subsequent hierarchy minimization) of equivalent cells during the hierarchical database construction phase of hierarchical Calibre applications.

Specifying YES instructs hierarchical Calibre applications to recognize cells that have different names but are otherwise identical. This can result in performance improvement in some designs because the number of unique cells may be greatly reduced. On the other hand, if previous stages have created huge cells, this statement may degrade performance. This statement is *not recommended* in new rule files.

Note

A potential drawback to specifying YES is that DRC errors may be underreported.

For example, if cells A and B are recognized to be the same, with subsequent replacement of B by A, then an error in cell A would not be seen as two errors (one in A and one in B) by the user. Hence, care must be used to fix errors that occur in duplicate cells. However, this is rarely a concern in post-tapeout flows where the output is GDS or OASIS format and containing assist features such as SRAFs.

This statement may be specified at most once.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Examples

```
//enable equivalent cell recognition
LAYOUT RECOGNIZE DUPLICATE CELLS YES
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

LAYOUT TURBO FLEX

SVRF statement

Specifies TURBOflex processing mode.

Usage

LAYOUT TURBO FLEX {<u>NO</u> | YES}

Arguments

- **<u>NO</u> | YES**

  Required keywords that indicate whether TURBOflex is to be used. Specify either NO or YES, but not both.

  NO represents the default behavior when the statement is not included in the rule file.

Description

An optional SVRF statement that specifies the rule file is run with TURBOflex. This statement may only appear once per rule file. If the statement appears additional times, the run exits with an error:

```
ERROR: Error SPC1 on line N of file - superfluous specification statement:
layout turbo flex.
```

This error message also appears if either of LAYOUT MONDO FLEX and LAYOUT ULTRA FLEX is also set to YES.

When TURBOflex mode is used, the transcript includes the line "TURBO FLEX ENABLED". Some things may cause TURBOflex mode to not be used even if this statement is in the rule file:

- If the environment variables CALIBRE_ULTRA_FLEX or CALIBRE_MONDO_FLEX are set.
- If the Calibre run is invoked with the -flat option. In that case, the Calibre Distributed Flat platform, which uses ULTRAflex, is used.
- If the Calibre run is invoked flat (that is, it does not include the -hier or -flat option), TURBOflex is not used.

Examples

LAYOUT TURBO FLEX YES

Related Topics

TURBOflex Data Construction

CALIBRE_TURBO_FLEX

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

LAYOUT ULTRA FLEX

SVRF statement

Specifies ULTRAflex processing mode.

Usage

LAYOUT ULTRA FLEX {<u>NO</u> | YES}

Arguments

- **<u>NO</u> | YES**

  Required keywords that indicate whether ULTRAflex is to be used. Specify either NO or YES, but not both.

  NO represents the default behavior when the statement is not included in the rule file.

Description

An optional SVRF statement that specifies the rule file is run with ULTRAflex. This statement may only appear once per rule file. If the statement appears additional times, the run exits with an error:

```
ERROR: Error SPC1 on line N of file - superfluous specification statement:
layout ultra flex.
```

This error message also appears if either of LAYOUT TURBO FLEX or LAYOUT MONDO FLEX is also set to YES.

When ULTRAflex mode is used, the transcript includes the line “ULTRA FLEX ENABLED”. If the Calibre run is invoked without either the -hier or -flat option, ULTRAflex is not used.

If the environment variable CALIBRE_MONDO_FLEX is set, MONDOflex is used instead of ULTRAflex.

Examples

LAYOUT ULTRA FLEX YES

Related Topics

ULTRAflex Data Construction

CALIBRE_ULTRA_FLEX

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

RET GLOBAL BOUND YES

SVRF statement

Applies a global bound on all unbounded SVRF operations in Calibre Distributed Flat or MDP Embedded mode.

Usage

RET GLOBAL BOUND YES

Arguments

None.

Description

This optional SVRF statement can be used instead of adding an explicit BOUND keyword on embedded SVRF operations. When RET GLOBAL BOUND YES is specified, an implicit bound of zero is applied on unlimited range operations that do not specify BOUND.

An unlimited range operation is an SVRF operation that requires a polygon’s full shape to be processed to generate the full result. For unlimited range operations to be used in section-based processing such as embedded SVRF, a range needs to be specified to determine what geometrical data from outside of a section's extent is included in the execution of a SVRF operation. The results of the unlimited range operation are accurate for the included geometrical data, but may not match the results if the polygon's full shape was included. Unlimited range operations require additional consideration and understanding of their supported functionality when used in embedded SVRF.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

RET GLOBAL LITHOFLAT YES

SVRF statement

Enables litho flat mode for all setlayer-based RET and LITHO commands.

Usage

RET GLOBAL LITHOFLAT YES

Arguments

None.

Description

This optional command activates ULTRAflex and litho flat processing mode on suitable rule files. When this command activates litho flat processing mode, the following line is added to the transcript:

```
LITHO-FLAT TURNED ON (automatically due to RET GLOBAL LITHOFLAT YES being
specified.)
```

See "Litho Flat Processing Mode" on page 35 for operations that are and are not supported.

Related Topics

ULTRAflex Data Construction

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Post-Tapeout Environment Variables

The post-tapeout environment variables are preferred by some, but Siemens EDA recommends using the equivalent SVRF commands instead for visibility. In case of a conflict between an environment variable and an SVRF setting, the environment variable's setting is used.

The font conventions used in the command descriptions are described in "Syntax Conventions" on page 12. Note that the environment variable names must be typed in all uppercase.

Table 5-2. Environment Variables for Post-Tapeout Flow Runs

Command	Description
CALIBRE_ENABLE_AVX	Enables the x86 instruction set architecture Advanced Vector Extensions (AVX), which can improve performance. Not recommended when the remote hosts include a mix of platforms.
CALIBRE_FS_DB	Stores intermediate data at the specified filer location instead of in memory. Requires Calibre FullScale.
CALIBRE_FS_ENABLE_CHECKPOINTING	Saves intermediate Calibre run data in a format that enables the run to be resumed after an interruption. Requires PFSDB.
CALIBRE_FS_HEARTBEAT_INTERVAL	Specifies how often to print an elapsed time message to indicate the Calibre primary process is still running.
CALIBRE_FS_MAX_READ_THREADS	Limits the number of concurrent threads that are reading the layer data when using Calibre FullScale.
CALIBRE_FS_MAX_WRITE_THREADS	Limits the number of concurrent threads that are writing to the output files when using Calibre FullScale.
CALIBRE_FS_PROGRESS_REPORTING_INTERVAL	Specifies how often to report PROGRESS messages in the main transcript even if the percentage has only slightly increased.
CALIBRE_FS_RDB_DISABLE_INDEX	Controls how Calibre FullScale writes DFM RDB rule check names, affecting presentation.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 5-2. Environment Variables for Post-Tapeout Flow Runs (cont.)

Command	Description
CALIBRE_FS_RESUME_SESSION	Restarts a Calibre FullScale run from a saved state.
CALIBRE_MONDO_FLEX	Sets the processing mode to MONDOflex.
CALIBRE_TURBO_FLEX	Sets the processing mode to TURBOflex.
CALIBRE_ULTRA_FLEX	Sets the processing mode to ULTRAflex.
LITHO_SOCS_KERNELS_SHARED	Causes CPUs on the same host machine to share a copy of the optical model SOCS directory, which can significantly reduce required memory.
LITHO_WRITE_DEBUG_SVRF	Generates additional information on LITHO operations that can be useful for creating test cases or debugging.
RET_WRITE_DEBUG_SVRF	Generates additional information on RET operations that can be useful for creating test cases or debugging.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

CALIBRE_ENABLE_AVX

Environment variable

Enables the x86 instruction set architecture Advanced Vector Extensions (AVX), which can improve performance. Not recommended when the remote hosts include a mix of platforms.

Usage

CALIBRE_ENABLE_AVX ON

Arguments

- **ON**

  A required argument that activates AVX extensions. By default, AVX is not used.

Description

This environment variable is ignored by the AOJ tree, which runs only on processors that support AVX2. AOJ always uses AVX2.

By default, AVX is not used. You must set this environment variable before launching a Calibre run to enable the use of AVX on compatible processors such as those that are built on the Intel Sandy Bridge and AMD Bulldozer architectures.

On processors that support AVX in their instruction sets, setting this variable can improve performance. Results will be slightly different than when AVX is not used, but no significant change in the resulting contours.

There are no issues when all remotes used in a run either support AVX or do not support AVX. However, when runs include a mix of AVX support there are occasionally problems at a tile boundary. If you see the warning “WARN: non-homogeneous litho fft settings are in use” and are not satisfied with results, disable the AVX extensions or set your cluster management software to use processors of a single type.

Note

Calibre nmSRAF gradient calculations are sensitive to the differences between AVX and non-AVX results. It is imperative for SRAFs that recipe development and production runs are done with identical AVX settings.

Examples

C Shell

To set:

```
% setenv CALIBRE_ENABLE_AVX ON
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

To unset:

```
% unsetenv CALIBRE_ENABLE_AVX
```

Bourne or Korn Shell

To set:

```
$ CALIBRE_ENABLE_AVX=ON
$ export CALIBRE_ENABLE_AVX
```

To unset:

```
$ unset CALIBRE_ENABLE_AVX
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

CALIBRE_FS_DB

Environment variable

Stores intermediate data at the specified filer location instead of in memory. Requires Calibre FullScale.

Usage

CALIBRE_FS_DB "*path*"

Arguments

- ***path***

  A required argument specifying the directory to use for the database. The directory should be on a file server partition that all hosts can access.

Description

This optional environment variable controls whether a Calibre FullScale run stores intermediate data in memory (the default) or uses files on disk. The disk method is referred to as persistent FullScale database (PFSDB).

PFSDB monitors access speed on the filer. It adds lines like the following to the transcript:

```
PROGRESS: 57%; LVHEAP = 2706/3625/3638 GEDB = 6822/11154 ELAPSED TIME =
5170 ERT = 1.9 HRS
<PFSDB: S 14.962 9.234 N 4959 26065 B 6752 1628 T 451283 174619 R 256 F
26383 5598 15475 D 635 911 2171>
```

where

- **S** indicates the read and write speed for the job in MB/sec. In this example, the filer is reading 14.962 MB/sec and writing at 9.234 MB/sec.
- **N** indicates the number of reads and writes in the run since the last report. In this example, it has read 4959 records and written 26065.
- **B** is the number of bytes read and written in MB since the last report. In this example, it has read 6752 MB and written 1628 MB.
- **T** is the elapsed time since the last report spent reading and writing. It is in milliseconds. In this example, it has spent 451283 ms (roughly 7.5 minutes) reading and 174619 ms (roughly 2.9 minutes) writing.
- **R** is the number of remotes. In this example, there were 256 remotes connected to the database on the filer.
- **F** is the file information. The first number is the total size of the database (26383 MB in the example), the number of records (5598), and the aggregate size of the files (15475 MB).

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- **D** is deletion information. The example shows 635 files deleted, which used 911 MB and took 2171 milliseconds to delete. The deletion time is reset after every report.

Because PFSDB is an alternative to RDS, do not include -remotedata when invoking.

Examples

C Shell

To set:

```
% setenv CALIBRE_FS_DB "/mnt/data/"
```

To unset:

```
% unsetenv CALIBRE_FS_DB
```

Bourne or Korn Shell

To set:

```
$ CALIBRE_FS_DB="/mnt/data"
$ export CALIBRE_FS_DB
```

To unset:

```
$ unset CALIBRE_FS_DB
```

Related Topics

Data Paths for Calibre FullScale

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

CALIBRE_FS_ENABLE_CHECKPOINTING

Environment variable

Saves intermediate Calibre run data in a format that enables the run to be resumed after an interruption. Requires PFSDB.

Usage

CALIBRE_FS_ENABLE_CHECKPOINTING TRUE

Arguments

- TRUE

  A required argument that enables saving application snapshots (checkpoints). Although this enables the feature, the information is not saved unless the run is also using PFSDB.

  By default, the temporary information the run writes to the file system is not in a form that a run can use for resuming.

Description

Calibre checkpoints provide a measure of fault tolerance. It is done by saving a snapshot of the run's state in a form that allows Calibre to load the state and continue execution without repeating the initial computations.

Because checkpoints require PFSDB and slightly more memory and processing time than standard runs, it is not recommended for all Calibre FullScale runs. Enable checkpointing for individual jobs that your cluster profiling indicates might not run to completion.

To save checkpoints, you must also have CALIBRE_FS_DB set. To resume from a checkpoint, set CALIBRE_FS_RESUME_SESSION to the most recent file beginning *CalibreFSState* in the PFSDB directory.

Examples

C Shell

To set:

```
% setenv CALIBRE_FS_DB "/mnt/data/"
% setenv CALIBRE_FS_ENABLE_CHECKPOINTING TRUE
```

To unset:

```
% unsetenv CALIBRE_FS_ENABLE_CHECKPOINTING
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Bourne or Korn Shell

To set:

```
$ CALIBRE_FS_DB="/mnt/data"
$ export CALIBRE_FS_DB
$ CALIBRE_FS_ENABLE_CHECKPOINTING=TRUE
$ export CALIBRE_FS_ENABLE_CHECKPOINTING
```

To unset:

```
$ unset CALIBRE_FS_ENABLE_CHECKPOINTING
```

Related Topics

Data Paths for Calibre FullScale

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

CALIBRE_FS_HEARTBEAT_INTERVAL

Environment variable

Specifies how often to print an elapsed time message to indicate the Calibre primary process is still running.

Usage

CALIBRE_FS_HEARTBEAT_INTERVAL *integer*

Arguments

- ***integer***

  A required value specifying how many minutes should pass between messages.

Description

For particularly large runs, it can be hard to tell whether the process on a primary host is active or has hung. When this variable is set, the primary process prints a message to the transcript at the specified interval, assuring administrators the run is still making progress.

By default, the heartbeat message of "CURRENT TIME = *timestamp* ELAPSED TIME = *seconds*" does not appear in the transcript. A progress report is printed roughly every 10 minutes, unless a different value is specified with CALIBRE_FS_PROGRESS_REPORTING_INTERVAL.

Examples

Following is a part of a transcript with the heartbeat interval set to 1, and the progress report interval set to 3. This results in a progress report about every three messages. The heartbeat messages are in bold font in the example.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

```
Processing tasks. LVHEAP = 9051/9053/9053  SHARED = 164/192  ELAPSED TIME
= 195 (2019/07/19 12:02:28)

PTO OP 1 (11/1/0/0)
        EMPTY: prev_sraf.1 no_bias_marker.1 prev_sraf no_bias_marker
mx.opc.e2_no.1 no_retarget_marker
CPU TIME = 0  OP TIME = 0  REMOTE REAL TIME = 0  OPS COMPLETE = 1 OF 437
ELAPSED TIME = 197 (2019/07/19 12:02:30)
CURRENT TIME = 2019-07-19 12:03:13  ELAPSED TIME = 240
CURRENT TIME = 2019-07-19 12:04:13  ELAPSED TIME = 300
CURRENT TIME = 2019-07-19 12:05:13  ELAPSED TIME = 360
PROGRESS: 0.02%; LVHEAP = 11332/11546/11575  SHARED = 690/704  GEDB =
65/526  ELAPSED TIME = 361  ERT = 19.7 HRS
    TMP<262>            : 84 / 84 MB +
    all_contact         : 83 / 83 MB +
    via_sized           : 81 / 81 MB +
    viabelow_all        : 43 / 43 MB +
CURRENT TIME = 2019-07-19 12:06:13  ELAPSED TIME = 420
CURRENT TIME = 2019-07-19 12:07:13  ELAPSED TIME = 480
CURRENT TIME = 2019-07-19 12:08:13  ELAPSED TIME = 540
PROGRESS: 0.02%; LVHEAP = 12833/12982/13034  SHARED = 1331/1344  GEDB =
530/1168  ELAPSED TIME = 545  ERT = 29.0 HRS
    TMP<262>            : 197 / 197 MB +
    all_contact         : 197 / 197 MB +
    via_sized           : 194 / 194 MB +
    viabelow_all        : 102 / 102 MB +
    TMP<51>             : 102 / 102 MB +
    viaabove_all        :  77 /  77 MB +
    TMP<60>             :  77 /  77 MB +
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

CALIBRE_FS_MAX_READ_THREADS

Environment variable

Limits the number of concurrent threads that are reading the layer data when using Calibre FullScale.

Usage

CALIBRE_FS_MAX_READ_THREADS *integer*

Arguments

- ***integer***

  A required value specifying the maximum number of threads for reading input.

Description

The CALIBRE_FS_MAX_READ_THREADS environment variable controls the *maximum* number of concurrent reads from the input. The default is the lesser of -turbo or 1000.

Overall performance suffers when too many threads attempt to read data simultaneously. The default value was chosen based on tests with varying hardware and layout sizes. Higher values may result in network congestion and overloading the file server.

This environment variable only controls threads attempting to read the input. See "CALIBRE_FS_MAX_WRITE_THREADS" on page 90 for writing output.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

CALIBRE_FS_MAX_WRITE_THREADS

Environment variable

Limits the number of concurrent threads that are writing to the output files when using Calibre FullScale.

Usage

CALIBRE_FS_MAX_WRITE_THREADS *integer*

Arguments

- ***integer***

  A required value specifying the maximum number of threads for writing output.

Description

The CALIBRE_FS_MAX_WRITE_THREADS environment variable controls the *maximum* number of concurrent writes to OASIS and ASCII. The default is the lesser of -turbo or 1000.

Overall performance suffers when too many threads attempt to write to a file simultaneously. The default value was chosen based on tests with varying hardware and layout sizes. Higher values may result in network congestion and overloading the file server.

This environment variable only controls threads attempting to write to a file. See “CALIBRE_FS_MAX_READ_THREADS” on page 89 for reading input.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

CALIBRE_FS_PROGRESS_REPORTING_INTERVAL

Environment variable

Specifies how often to report PROGRESS messages in the main transcript even if the percentage has only slightly increased.

Usage

CALIBRE_FS_PROGRESS_REPORTING_INTERVAL ***integer***

Arguments

- ***integer***

  A required value specifying how many minutes should pass between progress reports. The default value is 10 minutes (10). The minimum value is 1.

Description

Calibre FullScale runs print a progress message to the transcript when a task is completed that increases the percentage reported by at least 1% since the last progress message. By default, if ten minutes pass without an increase of at least 1%, another progress message is output showing the same percentage.

With CALIBRE_FS_PROGRESS_REPORTING_INTERVAL, you can set the time between messages to be shorter or longer. The message is printed when a task completes and it has been at least the specified number of minutes.

Examples

Following is part of a transcript when CALIBRE_FS_PROGRESS_REPORTING_INTERVAL was set to 1. Notice the ELAPSED time different between each is 61 seconds, as the message is only printed when a task completes. During the four minutes, the progress proceeds 0.45%, which normally would not generate a progress report.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

```
PROGRESS: 1%; LVHEAP = 14247/14661/14694  SHARED = 2538/2880  GEDB = 2200/
2375  ELAPSED TIME = 1110  ERT = 4.4 HRS
    TMP<59>             : 109 / 149 MB -
    TMP<50>             :  90 / 132 MB -
    line_end            :  33 /  33 MB +
    outside_edge_result :  22 /  22 MB +
    TMP<54>             :   7 /   7 MB +
    TMP<44>             :   6 /   6 MB +
    TMP<49>             :   3 /   3 MB +
    TMP<58>             :   3 /   3 MB +
PROGRESS: 1.03%; LVHEAP = 14205/14661/14694  SHARED = 2483/2880  GEDB =
1958/2322  ELAPSED TIME = 1171  ERT = 5.3 HRS
    TMP<59> : 69 / 149 MB -
    TMP<50> : 59 / 132 MB -
    TMP<54> : 13 /  13 MB +
    TMP<44> : 12 /  12 MB +
    TMP<49> :  6 /   6 MB +
    TMP<58> :  5 /   5 MB +
PROGRESS: 1.06%; LVHEAP = 14221/14661/14694  SHARED = 2423/2880  GEDB =
1950/2260  ELAPSED TIME = 1232  ERT = 6.1 HRS
    TMP<59> : 24 / 149 MB -
    TMP<54> : 21 /  21 MB +
    TMP<50> : 19 / 132 MB -
    TMP<44> : 19 /  19 MB +
    TMP<49> : 10 /  10 MB +
    TMP<58> :  7 /   7 MB +
PROGRESS: 1.41%; LVHEAP = 13979/15141/15142  SHARED = 2309/2880  GEDB =
1850/2145  ELAPSED TIME = 1293  ERT = 5.5 HRS
    not_inside_layer.1 : 33 / 33 MB +
    TMP<54>            : 31 / 31 MB +
    TMP<44>            : 26 / 26 MB +
    TMP<49>            : 12 / 12 MB +
    TMP<58>            : 10 / 10 MB +
    line_end_adj       :  4 /  4 MB +
    para_enc_1.1       :  2 /  2 MB +
    line_end_enc_1     :  1 /  1 MB +
PROGRESS: 1.45%; LVHEAP = 14006/15141/15142  SHARED = 2331/2880  GEDB =
1850/2168  ELAPSED TIME = 1354  ERT = 6.9 HRS
    line_end_adj   : 35 / 35 MB +
    para_enc_1.1   : 11 / 11 MB +
    line_end_enc_1 : 10 / 10 MB +
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

CALIBRE_FS_RDB_DISABLE_INDEX

Environment variable

Controls how Calibre FullScale writes DFM RDB rule check names, affecting presentation.

Usage

CALIBRE_FS_RDB_DISABLE_INDEX [FALSE | TRUE]

Arguments

- FALSE

  Restores the default behavior. When the variable is set to FALSE, Calibre FullScale runs produce output as though the variable was not set.

- TRUE

  An optional keyword to make the setting explicit. The commands “setenv CALIBRE_FS_RDB_DISABLE_INDEX TRUE” and “setenv CALIBRE_FS_RDB_DISABLE_INDEX” have the same result.

Description

By default, DFM RDB output from a Calibre FullScale run separates the results of rule checks from different sections using an index number. This causes the Calibre® RVE™ tree view to show the results for each section as separate groups. To instead show the results of a rule check as a single group, set CALIBRE_FS_RDB_DISABLE_INDEX.

The index corresponds to “_*m*” for the rulecheck_name parameter described in “Single Layer RDB File Format” in the *Calibre® YieldAnalyzer and YieldEnhancer User’s and Reference Manual*. See “Example 2 — Change to Output” on page 94.

This variable only affects output from the DFM RDB operation. It does not change the format of results databases (RDBs) from other operations such as DRC RESULTS DATABASE or DENSITY.

See “Calibre FullScale SVRF Support” on page 45 for information on DFM RDB when run with Calibre FullScale.

Examples

Example 1 — Setting

C Shell

The following examples show how to set and unset the variable using explicit values.

To set:

```
% setenv CALIBRE_FS_RDB_DISABLE_INDEX TRUE
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

To restore default behavior:

```
% setenv CALIBRE_FS_RDB_DISABLE_INDEX FALSE
```

Bourne or Korn Shell

These examples show how to set and unset the variable without using explicit values. The variable must be set before starting the run.

To set:

```
$ export CALIBRE_FS_RDB_DISABLE_INDEX
```

To unset:

```
$ unset CALIBRE_FS_RDB_DISABLE_INDEX
```

Example 2 — Change to Output

Given the following rule checks, the DFM RDB produces data as shown:

```
out1 { DFM RDB out1 "out.rdb" }
out2 { DFM RDB "out.rdb" CHECKNAME ErrorOut2 }
```

Default Output	With CALIBRE_FS_RDB_DISABLE_INDEX
`out1::<1>_3`	**`out1::<1>`**
`1 1 2 Thu Feb 20 17:47:48 2020`	`1 1 2 Thu Feb 20 17:47:46 2020`
`CELL FS_1 1 0 0 1 0 0`	`CELL FS_1 1 0 0 1 0 0`
`IL: out1`	`IL: out1`
`p 1 4`	`p 1 4`
`1385 3423`	`1385 3423`
`1462 3445`	`1462 3445`
`1462 3547`	`1462 3547`
`1385 3569`	`1385 3569`
`ErrorOut2_8`	**`ErrorOut2`**
`1 1 2 Thu Feb 20 17:47:48 2020`	`1 1 2 Thu Feb 20 17:47:46 2020`
`CELL FS_6 1 0 0 1 0 0`	`CELL FS_6 1 0 0 1 0 0`
`IL: out2`	`IL: out2`
`p 1 4`	`p 1 4`
`2387 3935`	`2387 3935`
`2452 3889`	`2452 3889`
`2544 3889`	`2544 3889`
`2609 3935`	`2609 3935`

The index is "_3" for the out1 block and "_8" for the ErrorOut2 block in the default output. (In a full RDB, there would be many out1 and ErrorOut2 blocks.) No other part of the output format is changed.

Related Topics

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

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

CALIBRE_FS_RESUME_SESSION

Environment variable

Restarts a Calibre FullScale run from a saved state.

Usage

CALIBRE_FS_RESUME_SESSION "*path*/CalibreFSState.*tag*"

Arguments

- **"*path*/CalibreFSState.*tag*"**

  A required argument specifying the full path to the file named *CalibreFSState.<tag>*. The ***path*** portion of the argument is the name of the PFSDB directory set in CALIBRE_FS_DB. The entire file path must be enclosed in quotation marks.

Description

Set this environment variable to resume a Calibre FullScale run from a checkpoint. Checkpoints are not created by default; CALIBRE_FS_ENABLE_CHECKPOINTING must be set and the run must use PFSDB.

When resuming a run, the new run does not need to use PFSDB. If it uses RDS, however, it does not save checkpoints of its own.

Runs can be resumed using updated software from the same quarterly release; that is, a run that was using Calibre version 2020.4_15.9 can be restarted with 2020.4_23.14, but not v2021.1 or v2020.3.

Examples

Consider a run started in a Bourne shell with these set:

```
CALIBRE_FS_DB="/mnt/data"
CALIBRE_FS_ENABLE_CHECKPOINTING=TRUE
```

If the run did not complete and the file */mnt/data/CalibreFSState.orw75.1468.2020_0830_1232* is the most recent CalibreFSState file, resume the run by first setting CALIBRE_FS_RESUME_SESSION and then running with Calibre FullScale:

```
CALIBRE_FS_RESUME_SESSION "/mnt/data/CalibreFSState.orw75.1468.2020_0830_1232"
export CALIBRE_FS_RESUME_SESSION

calibre -pto -turbo rules.svrf
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

CALIBRE_MONDO_FLEX

Environment variable

Sets the processing mode to MONDOflex.

Usage

CALIBRE_MONDO_FLEX [*integer*]

Arguments

- *integer*

  An optional value specifying the grid size. The default is 128.

  Note
  Be careful when using this argument, as it can degrade performance.

  Values below 16 result in a 16x16 grid. Values above 1024 result in a 1024x1024 grid. Values between 16 and 1024 are rounded to the nearest multiple of 16.

Description

The environment variable CALIBRE_MONDO_FLEX has precedence over the SVRF statement LAYOUT MONDO FLEX and both means of setting TURBOflex and ULTRAflex.

Examples

C Shell

```
% setenv CALIBRE_MONDO_FLEX
```

Bourne or Korn Shell

```
$ CALIBRE_MONDO_FLEX
$ export CALIBRE_MONDO_FLEX
```

Related Topics

LAYOUT MONDO FLEX

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

CALIBRE_TURBO_FLEX

Environment variable

Sets the processing mode to TURBOflex.

Usage

CALIBRE_TURBO_FLEX [*integer*]

Arguments

- *integer*

  An optional value specifying the grid size. The default is 64.

  Note
  Be careful when using this argument, as it can degrade performance.

  Values below 16 result in a 16x16 grid. Values above 1024 result in a 1024x1024 grid. Values between 16 and 1024 are rounded to the nearest multiple of 16.

Description

The environment variable CALIBRE_TURBO_FLEX has precedence over the LAYOUT … FLEX SVRF commands. Both CALIBRE_MONDO_FLEX and CALIBRE_ULTRA_FLEX have higher precedence than CALIBRE_TURBO_FLEX.

Examples

C Shell

```
% setenv CALIBRE_TURBO_FLEX
```

Bourne or Korn Shell

```
$ CALIBRE_TURBO_FLEX
$ export CALIBRE_TURBO_FLEX
```

Related Topics

LAYOUT TURBO FLEX

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

CALIBRE_ULTRA_FLEX

Environment variable

Sets the processing mode to ULTRAflex.

Usage

CALIBRE_ULTRA_FLEX [*integer*]

Arguments

- *integer*

  An optional value specifying the grid size. The default is 64.

  Note
  Be careful when using this argument, as it can degrade performance.

  Values below 16 result in a 16x16 grid. Values above 1024 result in a 1024x1024 grid. Values between 16 and 1024 are rounded to the nearest multiple of 16.

Description

The CALIBRE_ULTRA_FLEX environment variable has precedence over CALIBRE_TURBO_FLEX and the LAYOUT … FLEX SVRF statements. CALIBRE_MONDO_FLEX takes priority over CALIBRE_ULTRA_FLEX.

Examples

C Shell

```
% setenv CALIBRE_ULTRA_FLEX
```

Bourne or Korn Shell

```
$ CALIBRE_ULTRA_FLEX
$ export CALIBRE_ULTRA_FLEX
```

Related Topics

LAYOUT ULTRA FLEX

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

LITHO_SOCS_KERNELS_SHARED

Environment variable

Causes CPUs on the same host machine to share a copy of the optical model SOCS directory, which can significantly reduce required memory.

Usage

LITHO_SOCS_KERNELS_SHARED {**0** | **1** | **2**}

Arguments

- **0** | **1** | **2**

  A required argument that controls whether to share the SOCS directory. By default, each core maintains its own copy.

  0 — Do not share the SOCS directory. This is the default behavior.

  1 — Use the improved algorithm, which has better memory handling when jobs are prematurely terminated.

  1 is the recommended setting

  2 — Use the original algorithm. This is equivalent to the pre-2020.3 setting of TRUE. If a cluster management system terminates processes with SIGKILL instead of SIGTERM, the remote host needs to be rebooted to recover the memory.

  The setting of “2” is provided for backwards compatibility only.

Description

The LITHO_SOCS_KERNELS_SHARED environment variable enables cores on the same host that are part of the same Calibre run to share the SOCS directory (a part of the optical model) instead of each core having its own copy. The memory savings is generally greatest when using advanced optical models, such as DDM.

Monitoring Memory Savings

To track the shared memory, add the keywords “virtual lithoshared” to the MONITOR REMOTE MEMORY and MONITOR LOCAL MEMORY statements in the configuration file. The log file will add a virtual memory section marked with V and a shared litho section marked with LS to the MON:M entries.

For example, the following MON:M statement shows the V section in orange and the LS in green:

```
<MON:M 25449 H 3 322 471 V 1043 808 767 LS 224 56 224>
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Virtual Memory (V Section)

The virtual memory section is shown because of the “virtual” keyword in the MONITOR statement. Virtual memory is physically a mix of RAM and disk. The numbers can be interpreted as follows (all measurements in Mb):

- The first number is the total virtual memory, accounting for both RAM and disk.
- The second number is the maximum amount of RAM memory used so far. It may go up during the run but does not decrease.
- The third number is the current amount of RAM in use, and fluctuates throughout the run. This is also referred to as “virtual RSS” or “resident memory”.

Litho Shared Memory (LS Section)

The litho shared memory section is shown because of the “lithoshared” keyword in the MONITOR statement. It reports on the memory in use by all CPUs on the host both for this process and any other processes that may be running. (Note that the best practice for all Calibre runs is to ensure no other user processes are running on the same hardware.)

- The first number is the total shared memory used by a process. It includes the memory reported by all cores attached to the process.
- The second number is the effective memory usage. This value takes into account that memory is shared. From the set

  ```
  LS 224 56 224
  ```

  you can deduce there are four cores attached to the process because 224/56 = 4.
- The third number is the total memory in use on the host. It includes memory not used by the Calibre process that the MONITOR statement is attached to.

Tip

If a run is following best practices, the first and third numbers in the LS section should always be the same.

Calculation

To calculate the percentage of memory that shared memory saves, use the following formula, where V*number* and LS*number* refer to the first, second, or third number in the V or LS section.

```
{1 - (V2 - LS1 + LS2)/V2} x 100
```

Examples

C Shell

To set:

```
% setenv LITHO_SOCS_KERNELS_SHARED 1
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

To unset:

```
% unsetenv LITHO_SOCS_KERNELS_SHARED
```

or

```
% setenv LITHO_SOCS_KERNELS_SHARED 0
```

Bourne or Korn Shell

To set:

```
$ LITHO_SOCS_KERNELS_SHARED=1
$ export LITHO_SOCS_KERNELS_SHARED
```

To unset:

```
$ unset LITHO_SOCS_KERNELS_SHARED
```

Related Topics

MONITOR LOCAL [Calibre Administrator's Guide]

MONITOR REMOTE [Calibre Administrator's Guide]

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

LITHO_WRITE_DEBUG_SVRF

Environment variable

Generates additional information on LITHO operations that can be useful for creating test cases or debugging.

Usage

LITHO_WRITE_DEBUG_SVRF *id* | *name*

Arguments

- ***id***

  A required argument that specifies the LITHO statement to which the variable applies. An ***id*** of "1" references the first LITHO statement executed, "2" would be the second, and so on. Set ***id*** to -2 to output files for all relevant operations if testcase size is not a concern.

  Either ***id*** or ***name*** is required, but they cannot be specified together.

- ***name***

  A required argument that specifies the ***name*** of a LITHO FILE statement. When this form is used, the operations that use the specified litho setup file are written out.

  Either ***id*** or ***name*** is required, but they cannot be specified together.

Description

Use this variable to create simpler test cases for specific LITHO operations. (For RET operations, see "RET_WRITE_DEBUG_SVRF" on page 108.) The rule file must be run with the -hier switch; the variable is not supported with -pto or flat runs.

You can specify that the files be written to a directory other than the current working directory with LITHO_WRITE_DEBUG_SVRF_DATA_DIR. Set this to an existing directory.

LITHO_WRITE_DEBUG_SVRF causes the Calibre run to create OASIS and SVRF debugging files. The filenames depend on how the variable is specified:

- If you set ***id*** to a positive number, the files have a base name of *litho_<operation>_debug* where <*operation*> is the second keyword in the SVRF operation (for example, denseopc from LITHO DENSEOPC).
- If you set ***id*** to -2 (echo all), the files have a base name of *litho_<operation>_<N>_debug* where <*N*> starts at 1 for the first LITHO operation.
- If you set ***name***, the files have a base name of *litho_<name>_<N>_debug* where name is the setup file you specified and <*N*> indicates the sequence through it.

The <*filename*>*.svrf* file references the <*filename*>*.oas* file. When the SVRF file is executed, it runs only the single LITHO operation.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

It is simpler to use the ***name*** argument instead of the ***id*** argument. The id argument requires you to count the operations executed. In most cases, you cannot rely on the “LITHO RUN” banner for determining the appropriate ID. The number shown in the banner increments for RET operations such as RET NMDPC but does not increment for sparse OPC such as LITHO OPCPRO.

Often several LITHO operations run concurrently. Concurrent operations count as one invocation, as shown in “Example 2 - Counting in Transcript” on page 104. To determine the value to use for ***id***:

1. Save the transcript from running the real Calibre job. The job does not need to complete but does need to start the operation(s) intended for separation.
2. In the transcript, move past the rule file compilation stage. One way to do this is to search for the string

   ```
   COMPILATION MODULE COMPLETED
   ```

3. Search for the string LITHO occurring during execution, and count the number of jobs being started. This gives the ***id***.
   - A job consists of any cluster of “*layer* = LITHO *tool*” lines appearing together. The clustering indicates they are being processed concurrently.
   - Each LITHO command in the SVRF file appears at least twice in the operation stage of the transcript: once when it starts being processed, and again when it ends. Only count the ones starting the job.
   - Do not count LITHO FILE statements, environment variables, or other LITHO occurrences.

   Note
   You can obtain complete test cases by setting ***id*** to -2. If test case size is not a concern, this is the most reliable method for creating a complete test case.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Examples

Example 1 - Counting in SVRF File

Assume you have an SVRF file containing the following statements:

```
LAYER metal1 11
LAYER metal3 15

opc_m1 = LITHO DENSEOPC FILE m1 metal1 MAP metal1_opc
opc_m1 {COPY opc_m1} DRC CHECK MAP opc_m1 oasis 111 m1_out.oas
opc_m3 {LITHO OPC metal3 FILE m3} DRC CHECK MAP opc_m3 m3_opc.gds

LITHO FILE m1 [...
   layer metal1 hidden atten 0.06
   denseopc_options opts {...
      layer metal1 opc atten 0.06
   }
   setlayer metal1_opc = denseopc metal1 MAP metal1 OPTIONS opts
]

LITHO FILE m3 [...
   layer 15 metal3 0 0 opc dark
]
```

If you do not set LITHO_WRITE_DEBUG_SVRF, your output is the files *m1_out.oas* and *m3_opc.gds*, and the usual result databases and reports.

To produce files for a Calibre OPCpro test case, set LITHO_WRITE_DEBUG_SVRF to 2 in the shell before invoking Calibre. (This points to the second LITHO command, on the line beginning “opc_m3”.) This generates the litho_opc_debug files in addition to the usual output.

Example 2 - Counting in Transcript

In the following transcript excerpt, setting LITHO_WRITE_DEBUG_SVRF to 2 would return a group of 3 concurrent operations. Setting LITHO_WRITE_DEBUG_SVRF to *m2_orc.in* would have the same effect.

LITHO operation invocations are in bold. Duplicate invocations are in green.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

```
Running Calibre: $CALIBRE_HOME/calibre -remotefile gridflex.cfg -64 -hyper
-turbo -turbo_litho -drc -hier rules

Environment variable LITHO_WRITE_DEBUG_SVRF (='2') is being used
//  Calibre v2021.1_33.19    Mon Mar 1 16:54:23 PST 2021
//  Calibre Utility Library   v0-10_13-2017-1    Wed Jun 3 07:42:12 PDT 2020
//  Litho Libraries v2021.1_33.19  Mon Mar 1 16:10:01 PST 2021
//
//                   Copyright Siemens 1996-2021
//                       All Rights Reserved.
//   THIS WORK CONTAINS TRADE SECRET AND PROPRIETARY INFORMATION
//      WHICH IS THE PROPERTY OF MENTOR GRAPHICS CORPORATION
//        OR ITS LICENSORS AND IS SUBJECT TO LICENSE TERMS.
//
//  The registered trademark Linux is used pursuant to a sublicense from LMI,
the
//  exclusive licensee of Linus Torvalds, owner of the mark on a world-wide
basis.
//
//  Mentor Graphics software executing under x86-64 Linux

...

Layer FOM::<1> DELETED -- LVHEAP = 349/415/415  SHARED = 121/192

WARNING: 20 polygon result(s) segmented in DRC RuleCheck M1.

M2_fill = LITHO OPC m2_opc diff FILE ./litho/m2_opc.in
-----------------------------------------------------------------------
Operation EXECUTING on HDB 0 (T506 S15 E0:383 H0:195 A16:644 L0:186 C5)

...

TIME FOR LITHO CORMERGE:       CPU TIME = 238 + 1687  REAL TIME = 158
TIMESTAMP 2276
<MON:E 2275 O 69>

M2_fill = LITHO OPC m2_opc diff FILE ./litho/m2_opc.in
-----------------------------------------------------------------------
M2_fill (HIER-PMF TYP=1 CFG=0 HGC=3259010...

...

Layer xG_67m.2h::<1> DELETED -- LVHEAP = 1284/1332/1332  SHARED = 970/992

m2_90::<1> = LITHO ORC m2_orc m2_targ MAPNUMBER 300 FILE ./litho/m2_orc.in
m2_85::<1> = LITHO ORC m2_orc m2_targ MAPNUMBER 301 FILE ./litho/m2_orc.in
m2_80::<1> = LITHO ORC m2_orc m2_targ MAPNUMBER 302 FILE ./litho/m2_orc.in
-----------------------------------------------------------------------
Operation EXECUTING on HDB 0 (T2441 S15 E0:472 H4:287 A0:947 L0:229 C5)

...

Layer v12 DELETED -- LVHEAP = 1750/3390/3393  SHARED = 1371/1696

Layer fin DELETED -- LVHEAP = 1750/3390/3393  SHARED = 1371/1696

m2_90::<1> = LITHO ORC m2_orc m2_targ MAPNUMBER 300 FILE ./litho/m2_orc.in
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

```
m2_85::<1> = LITHO ORC m2_orc m2_targ MAPNUMBER 301 FILE ./litho/m2_orc.in
m2_80::<1> = LITHO ORC m2_orc m2_targ MAPNUMBER 302 FILE ./litho/m2_orc.in
-----------------------------------------------------------------------
m2_90::<1> (HIER TYP=1 CFG=1 HGC=0 FGC=0 HEC=0 FEC=0 VHC=F VPC=F)

...

Layer cp1m.orc.2: p1m.pinch ::<1> DELETED -- LVHEAP = 3026/3871/3872  SHARED =
2809/3584

via_Mopc = LITHO OPC via_opcIn FILE ./litho/vim/setup/via_opc.in
----------------------------------------------------------------
Operation EXECUTING on HDB 0 (T3371 S15 E0:476 H2:450 A0:970 L0:229 C3)

...

via_Mopc = LITHO OPC via_opcIn FILE ./litho/vim/setup/via_opc.in

...
```

Example 3 - Filenames

When developing Calibre pxOPC recipes, a fairly standard procedure is to generate a curved target with a LITHO OPCVERIFY call, run Calibre pxOPC using RET PXOPC, and then check the results with a second LITHO OPCVERIFY call. This produces a transcript with the following banners:

```
==============================================================================
=====                START OPCVERIFY EXECUTION (LITHO RUN 1)             =====
==============================================================================
------------------------------------------------------------------------------
-----                             SETUP FILE                             -----
------------------------------------------------------------------------------
--- OPCVERIFY SETUP FILE = smooth.in

...
==============================================================================
=====                  START PXOPC EXECUTION (LITHO RUN 2)               =====
==============================================================================
------------------------------------------------------------------------------
-----                             SETUP FILE                             -----
------------------------------------------------------------------------------
--- PXOPC SETUP FILE = pxopc.in

...
==============================================================================
=====                START OPCVERIFY EXECUTION (LITHO RUN 3)             =====
==============================================================================
------------------------------------------------------------------------------
-----                             SETUP FILE                             -----
------------------------------------------------------------------------------
--
--- OPCVERIFY SETUP FILE = opcv
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

The simplest method to be sure you have the correct operation is to use the setup file name. Before running again, set LITHO_WRITE_DEBUG_SVRF to the appropriate setup file name. For example:

```
setenv LITHO_WRITE_DEBUG_SVRF opcv
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

RET_WRITE_DEBUG_SVRF

Environment variable

Generates additional information on RET operations that can be useful for creating test cases or debugging.

Usage

RET_WRITE_DEBUG_SVRF *id* | *name*

Arguments

- ***id***

  A required argument that specifies the RET operation to which the variable applies. An ***id*** of "1" references the first RET operation executed, "2" would be the second, and so on. Set ***id*** to -2 to output files for all relevant operations.

  Either ***id*** or ***name*** is required, but they cannot be specified together.

- ***name***

  A required argument that specifies the ***name*** of a LITHO FILE statement. When this form is used, the operations that use the specified litho setup file are written out.

  Either ***id*** or ***name*** is required, but they cannot be specified together.

Description

Use this variable to create simpler test cases for the RET operations such as RET DPSTITCH and RET MBSRAF. The rule file must be run with the -hier switch; the variable is not supported with -pto or flat runs.

RET_WRITE_DEBUG_SVRF causes the Calibre run to create OASIS and SVRF debugging files. The filenames depend on how the variable is specified:

- If you set ***id*** to a positive number, the files have a base name of *ret_<operation>_debug* where <*operation*> is the second keyword in the SVRF operation (for example, REFINE from RET REFINE).
- If you set ***id*** to -2 (echo all), the files have a base name of *ret_<operation>_<N>_debug* where <*N*> starts at 1 for the first RET operation.
- If you set ***name***, the files have a base name of *ret_<name>_<N>_debug* where name is the setup file you specified and <*N*> indicates the sequence through it.

Examples

For a run that included both RET PXOPC and LITHO OPCVERIFY, setting both LITHO_WRITE_DEBUG_SVRF and RET_WRITE_DEBUG_SVRF to -2 resulted in the following debug files:

- *litho_opcverify_1_debug.svrf*

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- *ret_PXOPC_1_debug.svrf*
- *litho_opcverify_1_debug.oas*
- *ret_PXOPC_1_debug.oas*

The two variables create files that are separately incremented.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Litho Setup File Commands

Although the commands that affect processing mode are located in the various products' litho setup files, all products within a single SVRF rule file must be set up to use the same mode.

For more on the processing modes, see "Direct Data Access" on page 24.

Table 5-3. Litho Setup File Commands for Processing Mode

Command	Description
direct_input	Enables direct data entry (DDE), which can improve run time in some cases. Requires litho flat processing mode. Forbidden in Calibre FullScale (-pto) rule files.
direct_output	Enables direct data output (DDO), which can improve run time in some cases. Requires litho flat processing mode. Forbidden in Calibre FullScale (-pto) rule files.
pixel_alignment_v2	Controls pixel alignment settings in litho flat processing mode.
processing_mode	Flattens geometries when the database was constructed hierarchically.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

direct_input

Type: litho setup file

Supported by: LITHO DENSEOPC, LITHO MASKOPT, LITHO MPC, LITHO NMBIAS, LITHO OPCVERIFY, and RET PXOPC

Enables direct data entry (DDE), which can improve run time in some cases. Requires litho flat processing mode. Forbidden in Calibre FullScale (-pto) rule files.

Usage

direct_input ‘**{**’
    { **vboasis_path** ***filename***
        {[input *input_number*] **layer** ***number*** [*datatype*]} …
        [vboasis_precision_multiplier {*n*/*d* | AUTO}]
        [direct_FS_access]
    } …

‘**}**’

Arguments

- **vboasis_path** ***filename***

  A required argument supplying the absolute path to an OASIS filename to be used as the design source. A runtime error is returned if the input file is invalid. The filename must not be in single or double quotes.

  Multiple **vboasis_path** sets can be specified in a direct_input statement.

- [input *input_number*] **layer** ***number*** [*datatype*]

  A required keyword set that specifies one or more layers. At least one layer declaration is required.

  input *input_number* — An optional keyword set that specifies which input layer from the setup file to map. The index starts at 1 for the first layer listed in the setup file. The specified layer is mapped to the VB:OASIS layer ***number***, allowing you to use the input layers out of order. The value you specify for *input_number* cannot exceed the number of setup layers. If this keyword set is not specified, layers are assumed to be in the same order as the setup file.

  Either all layers must use input *input_number*, or none of them.

  layer ***number*** — A required keyword set defining a layer that is present in ***filename***.

  datatype — An optional argument that limits the datatypes that map onto the post-tapeout operation’s layers. If *datatype* is absent, all datatypes for that layer are used.

- vboasis_precision_multiplier {*n*/*d* | AUTO}

  An optional argument that modifies the PRECISION of ***filename***. It may be specified once per vboasis_path. Either AUTO or a ratio must be specified.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

n/*d* — Multiplies the PRECISION by *n/d*, and then scales the data from the file by the same amount to retain the same absolute scale as the original PRECISION setting.

AUTO — Attempt to infer the correct ratio. AUTO exits from the parser with an error if the ratio is not a rational number or would cause arithmetic overflow.

- direct_FS_access

  An optional keyword that causes the remotes to read ***filename*** without going through the primary host.

Description

This optional command enables direct data entry (DDE). It works in conjunction with litho flat processing mode and automatically activates it when a direct_input block is found, even if processing_mode is set to hierarchical.

This statement can appear only once per litho setup file. Use multiple **vboasis_path** sets in a single statement to access multiple files.

This command causes the layers in LAYOUT PATH to be left unused. They are replaced with layers read directly from the OASIS database specified with the **vboasis_path** argument. If direct_output is not also specified, only the OASIS data within the coordinates of the HDB extent is directly read. Therefore, it is not enough to pass in dummy layers through the calling LITHO, RET, or MDP statement; at least one DRC RDB layer must contain a square large enough to cover the given OASIS file in order to ensure that everything in it gets processed.

There is a one-to-one mapping between the layers specified in this command and the layers specified using the layer command. This mapping is based on the layer order.

Examples

The following example of an SVRF file with a single Calibre pxOPC call declares an empty OASIS file for the traditional (HDB) input, and then directly reads the *test.oas* and *test2.oas* OASIS files.

```
LAYOUT PATH "dummy.oas"

LAYER layout_window 1
LAYER dummy1 998
LAYER dummy2 999

# Write layout_window to DRC RDB to ensure the OASIS data is read
layout_window { COPY layout_window } DRC CHECK MAP layout_window 1000

M1 = RET PXOPC layout_window dummy1 dummy2 FILE setup MAP M1_copy
M1_OPC = RET PXOPC layout_window dummy1 dummy2 FILE setup MAP M1_OPC
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

```
LITHO FILE setup [ /*

   direct_input {
      vboasis_path test.oas
         input 1 layer 43
         input 3 layer 46
      vboasis_path “test2.oas”
         input 2 layer 33 100
   }

   modelpath models
   optical_model_load op_mod
   resist_model_load res_model

   processing_mode flat     # Optional when using direct_input.

   layer M1 visible clear   # Without direct_input, this would have been
                            # "layout_window." With direct_input this is
                            # layer 43 in the file "test.oas".

   layer SRAF visible clear # Without direct_input, this would have been
                            # "dummy1." With direct_input this is layer
                            # 33, datatype 100, in the file "test2.oas".

   layer CONT hidden clear  # Without direct_input, this would have been
                            # "dummy2." With direct_input this is layer
                            # 46 in the file “test.oas”.

   pxopc_options px.opt {
   ...
   }

setlayer M1_copy = PXOPC ...
setlayer M1_OPC = PXOPC ...
*/]
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

direct_output

Type: litho setup file

Supported by: LITHO DENSEOPC, LITHO MASKOPT, LITHO MPC, LITHO NMBIAS, LITHO OPCVERIFY, and RET PXOPC

Enables direct data output (DDO), which can improve run time in some cases. Requires litho flat processing mode. Forbidden in Calibre FullScale (-pto) rule files.

Usage

direct_output ‘{’
  {**vboasis_path *filename*** [append]
    {**output *name* layer *number*** [*datatype*]}…
    [vboasis_output_primary *source*] } …
  ‘}’

Arguments

- **vboasis_path *filename*** [append]

  A required argument supplying the absolute path to an OASIS filename. All remote hosts must be able to access the file.

  Multiple **vboasis_path** sets can be specified in a direct_output statement.

  Use the optional append keyword if ***filename*** already exists. This causes the output to be added to ***filename*** rather than overwriting it.

- **output *name* layer *number*** [*datatype*]

  A required keyword set that specifies one or more layers. At least one layer declaration is required.

  output *name* — A required argument specifying a layer name created with a setlayer command elsewhere in the Litho setup file.

  layer *number* — A required argument specifying the layer number to use when writing to ***filename***.

  datatype — An optional argument that can be used to specify a datatype sublayer.

- vboasis_output_primary *source*

  An optional argument that can be specified once per direct_output command. It specifies the source of the top cell name in ***filename***. Use one of the following:

  HDB — The topcell name from the HDB.

  DIRECTIN — The top cell name from the first **vboasis_path** input.

  FILE — Same as DIRECTIN.

  explicit *name* — Explicitly names the top cell.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Description

This optional command enables direct data output (DDO). To prevent the compiler optimizing away “empty” layers, layers written to ***filename*** must also be passed to the SVRF level with the operation’s MAP keyword and written to a results database.

The direct_output command works in conjunction with litho flat processing mode and automatically activates it. You must set the CALIBRE_MTFLEX_LOCAL_HOST_DIR environment variable and the LOCAL HOST DIR configuration statement, as described in the *Calibre Administrator’s Guide*.

This statement can appear only once per litho setup file. Use multiple **vboasis_path** arguments in a single statement to output to multiple files.

Examples

The example SVRF file combines direct_input and direct_output. It shows the use of fake input and output files (*dummy_in.oas* and *dummy_out.oas*) and layer mapping.

Example 5-1. Direct Data Instead of HDB

```
LAYOUT PATH "dummy_in.oas"
LAYER target 5 100
LAYER contour 1006

DRC RESULTS DATABASE "dummy_out.oas" OASIS
POLYGON 0 0 1000 1000 target
LAYOUT ULTRA FLEX YES

X = RET PXOPC target contour MAP final FILE setup.in
pxopc_out { COPY X }
DRC CHECK MAP pxopc_out 101 2

LITHO FILE setup.in [
   direct_input {
    vboasis_path "/net/machine1/export/input.oas"
    input 1 layer 5 100
    input 2 layer 1006
   }

   layer target hidden clear
   layer contour hidden clear

   pxopc_options opt [
      ...
      layer target correction
      ...
   ]

   setlayer final = pxopc target contour MAP target OPTIONS opt
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

```
        direct_output {
         vboasis_path "/net/machine1/export/output.oas"
         output final layer 101
       }
     ]
     ...
```

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

pixel_alignment_v2

Type: litho setup file

Supported by: Operations that can use setlayer commands in the setup file.

Controls pixel alignment settings in litho flat processing mode.

Usage

pixel_alignment_v2 {**on** | **off**}

Arguments

- **on** | **off**

  Required arguments that indicate whether the pixel alignment settings introduced in version 2018.4 are to be used. The settings are on by default.

Description

When the optical image grid is not a multiple of the resist image grid, fractional upsampling occurs, which can cause differences between full chip and clipped runs in some cases. The default setting (equivalent to pixel_alignment_v2 on) corrects this issue.

Note

The ability to turn pixel_alignment_v2 off is provided solely for acceptance testing. It is not recommended for production-level setup files.

Related Topics

imagegrid aerial [Calibre OPCverify User's and Reference Manual]

simulation_consistency [Calibre nmOPC User's and Reference Manual]

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

processing_mode

Type: litho setup file

Supported by: Operations that can use setlayer commands in the setup file.

Flattens geometries when the database was constructed hierarchically.

Usage

processing_mode {**hierarchical** | **flat** | **hybrid**}

Arguments

- **hierarchical**

  A required argument that specifies the default mode of processing input. In this mode, the layer is split into cells and then tiles. This is more efficient when the layout is extremely hierarchical.

- **flat**

  A required argument that turns on litho-flat processing. This has the following limitations:

  - The amount of data generated may be larger because of pushing flat output into a hierarchical database. This could slow down subsequent SVRF operations.
  - When using flat mode, the run that creates the HDB must use ULTRA FLEX processing or the calling operation terminates with an error.

  In litho-flat mode, the full chip is split into tiles, which are then processed separately on remote machines. This is more efficient than hierarchical mode when the layout is almost flat.

- **hybrid**

  A required argument that analyzes the input design hierarchy. Cells containing significant hierarchy are processed in hierarchical mode, while the rest is processed in litho-flat mode. If the top-level cell is deemed sufficiently hierarchical, this mode is equivalent to **hierarchical**.

  This mode has the following limitations:

  - It is only supported for LITHO DENSEOPC, LITHO OPCVERIFY, LITHO NMBIAS, OPCBIAS, OPCSBAR, RET MBSRAF, and RET SBAR.
  - Because there is no guarantee that cells will use litho-flat processing, hybrid mode cannot be used with Calibre Distributed Flat processing.
  - When using hybrid mode, the run that creates the HDB must use ULTRA FLEX or TURBO FLEX processing.

Description

An optional command that must be used when the run is invoked with calibre -drc -hier. When flat Calibre nmDRC is invoked, this command is ignored.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

The processing_mode setting is also ignored when using direct_input or direct_output.

Related Topics

RET GLOBAL LITHOFLAT YES

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Chapter 6
Calibre Post-Tapeout Flow Best Practices

This chapter contains the following sections:

General Best Practices

Following is a checklist list of best practices, primarily for runtime. Follow the links for more detailed descriptions of the subjects.

Table 6-1. Post-Tapeout Flow Recommendations

Recommendation	**Notes**
Recent Calibre version	A Calibre quarterly version less than one year old is recommended to take advantage of continuous performance improvements.
do not use SIGKILL	When you need to terminate a Calibre process, use SIGTERM to exit cleanly. The SIGKILL signal forces immediate program termination and may not free memory or process table entries.
LAYOUT BASE LAYER	Provides more robust construction, especially important for a production PTO environment. Choice of base layers is application specific. Generally, fewer base layers are recommended with a large processing cluster.
LAYOUT RENAME ICV YES	Prevents data collisions when importing multiple layout files that may contain ICV cells.
LAYOUT TURBO FLEX YES	Provides a more robust construction for a production PTO environment.
Hyperscaling	Provides significant scaling improvements for non-litho operations.
do not use PUSH	The PUSH command is no longer generally recommended for reducing runtime. An exception is for dummy fill layers that may be passed whole or in part to Litho or RET operations. In these cases there may be some runtime benefit from doing a PUSH on the input fill layer to the target layer. Always evaluate the benefit to the overall runtime.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 6-1. Post-Tapeout Flow Recommendations (cont.)

Recommendation	Notes
OASIS	Smaller file size over GDSII, and faster data transfer.
OASIS CBLOCK	Smaller OASIS file size.
OASIS STRICT	Faster file access for subsequent disk-based processing (viewing and MDP EMBED).
DRC MAXIMUM VERTEX	Default of 4096 is recommended. Smaller values cause longer runtime for polygon output, and input in subsequent jobs.
LAUNCH CLUSTER	Parallel connection can save significant runtime for hundreds or thousands of remote hosts.
sufficient memory	Hardware memory should be evaluated against application-specific requirements to avoid swapping.
local disk	Network access and contention can increase runtime dramatically. A local disk is especially recommended for some disk intensive operations (primarily MDP).
tilemicrons (nmOPC and OPCverify)	Runtime for tiled LITHO operations is especially dependent on optimizing the tilemicrons value. Guidelines are given per technology node.
LITHO NMBIAS	nmBIAS is a tiled replacement for OPCbias, which can give better runtime performance, especially on larger clusters.
MONITOR	Provides more detailed runtime data, to facilitate analysis and improvement.
progress_meter (nmOPC and OPCverify)	Provides metrics for analysis and projection of runtime.

Related Topics

Optimal Performance Using LAYOUT BASE LAYER

Which Construction Method to Use

Running MT and Calibre MTflex with Hyperscaling [Calibre Administrator's Guide]

Output of OASIS CBLOCK Records and Strict Mode [Open Artwork System Interchange Standard (OASIS)]

Memory Usage Best Practices [Calibre Administrator's Guide]

Tool-Specific Best Practices

Many of the Calibre manuals include a chapter on best practices that document recommended settings and techniques for improving quality or runtime.

The following list is arranged by tool.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- "Hardware Configuration Guidelines" and "Network Configuration Guidelines" in the *Calibre Administrator's Guide*.
- "MDP Best Practices" in the *Calibre Mask Data Preparation User's and Reference Manual*.
- "Calibre Multi-Patterning Best Practices" chapter in the *Calibre Multi-Patterning User's and Reference Manual*.
- "Calibre nmOPC Best Practices" chapter (especially "Recommended Settings for Best Runtime") in the *Calibre nmOPC User's and Reference Manual*.
- "Calibre nmSRAF Best Practices" chapter in the *Calibre nmSRAF User's and Reference Manual*.
- "Calibre OPCpro Best Practices" chapter in the *Calibre OPCpro User's and Reference Manual*.
- "Calibre OPCsbar Best Practices and Problem Resolution" chapter in the *Calibre OPCsbar User's and Reference Manual*.
- "Calibre LPE Best Practices" chapter in the *Calibre Local Printability Enhancement User's and Reference Manual*.
- "Calibre pxOPC Best Practices" in the *Calibre pxOPC User's and Reference Manual*.
- "Best Practices" section in the *Calibre OPCverify User's and Reference Manual*.
- "Best Practices for Calibre RET Modeling" appendix in the *Calibre WORKbench User's and Reference Manual*.
- "Optimal Performance Using LAYOUT BASE LAYER" (especially "How to Tell Which Layers are Base Layers") in this manual.

SVRF Optimization in the PTO Flow

Following is a checklist of SVRF optimizations that can reduce runtime. There are commonly multiple methods to get the same results in Calibre. Some have significant differences in runtime or hierarchical degradation.

For more details, see these sections in the *Calibre Verification User's Manual*:

- Maximizing DRC Capacity and Minimizing Run Time
- Hierarchical Operation Efficiency

Table 6-2. SVRF Recommendations

Recommendation	Notes
Concurrency	Where possible, use SVRF concurrency to reduce runtime.
Hierarchy	Where possible, choose operations that preserve hierarchy.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 6-2. SVRF Recommendations (cont.)

Recommendation	Notes
GROW, SHRINK	Avoid; inherently destroys hierarchy.
SHIFT	Avoid; inherently destroys hierarchy.
ANGLE	Avoid the constraints Angle == 0 and Angle == 90; they inherently destroy hierarchy.
HOLES	Avoid; inherently destroys hierarchy.
PERIMETER	Avoid; inherently destroys hierarchy.
VERTEX	Avoid; inherently destroys hierarchy.
CONNECT	Avoid; tends to destroy hierarchy and also reduces scaling on large clusters.
NOTCH, SPACE	Avoid; noticeably hierarchical runs.
Large Constraints	Avoid very large values in EXT or ENC operations. These will tend to destroy hierarchy and also reduces scaling on large clusters. Use EXPAND EDGE or SIZE instead.
Dimensional Checks	Where possible, avoid dimensional checks. Where dimensional checks must be used, reduce the layer data. One method of reducing layer data is conjunctive checks.
Runtime TVF	Avoid runtime TVF. Hyperscaling processes SVRF more efficiently than runtime TVF.

Related Topics

Concurrency [Calibre Verification User's Manual]

Hierarchy Preservation by Layer Operations [Calibre Verification User's Manual]

Efficient Width and Spacing Checks [Calibre Verification User's Manual]

Conjunctive Checks [Calibre Verification User's Manual]

Runtime TVF [Standard Verification Rule Format (SVRF) Manual]

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Index

Third-Party Information

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

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.