LVR User Guide

Layout Verification Reporter
Physical verification for multi-million gate designs

LVR 2.0 · Documentation Rev A
August 2026

Ramtera Design Automation, Inc.
Bangalore, India | Lewes, Delaware, USA
www.ramtera.com | support@ramtera.com

Legal Notice

Ramtera Design Automation, Inc.
16192 Coastal Highway, Lewes, Delaware 19958, County of Sussex, USA
www.ramtera.com | support@ramtera.com

Revision History

Revision	Date	Product	Summary
Rev A	August 2026	2.0	First release of the consolidated user guide. Adds installation and platform support, per-flow chapters for every engine, migration guidance, performance tuning, troubleshooting and the diagnostic message reference.

This document describes LVR 2.0 (build 2.0.0). Verify against the release notes for the build you are running.

Contents

1 Introduction

LVR — Layout Verification Reporter — is a physical verification platform covering design rule checking, layout-versus-schematic comparison, electrical rule checking, layout differencing, density and manufacturability analysis, antenna checking and parasitic extraction from a single engine and a single rule deck.

LVR reads SVRF natively. A foundry deck written for an existing sign-off tool runs without translation, which means adopting LVR does not require rewriting or maintaining a second copy of the rule set.

1.1 Engines

Mode	Engine	Compares	Primary output
`DRC`	Design rule check	Layout against a rule deck	Geometry violation markers
`LVS`	Layout versus schematic	Extracted layout netlist against a source netlist	Device and net mismatches, shorts, opens
`MLVS`	Macro LVS	Top level against macro pin interfaces	Macro-level connectivity mismatches
`SVS`	Schematic versus schematic	Two netlists, no layout	Netlist differences
`ERC`	Electrical rule check	Extracted connectivity against electrical rules	Floating gates, supply shorts, undriven nets
`XOR`	Layout difference	Two layouts, exactly	Difference regions
`FASTXOR`	Screening difference	Two layouts, approximately	Difference indication
`DENSITY`	Density check	Fill ratio per window against rule limits	Out-of-range window markers
`DFM`	Manufacturability	Layout against recommended rules	Advisory markers
`ANTENNA`	Antenna ratio	Accumulated conductor area per gate	Antenna violations
`PEX`	Parasitic extraction	Layout geometry to an RC network	SPEF, DSPF, SDF, IR and EM reports

1.2 Architecture

LVR partitions the layout into regions and processes them in parallel across a worker pool. Each worker owns a region and evaluates the full rule set within it; interactions crossing a region boundary are resolved through a halo band that each worker reads beyond its own edge. Scaling is close to linear until the design stops partitioning evenly.

Geometry is handled at the lowest tier that the design requires. Rectilinear layouts use the Manhattan-only representation, which is substantially faster; layouts containing 45-degree structures such as crackstops use the 45-degree tier; arbitrary-angle geometry falls to the general tier. Choosing a tier higher than the design needs is the single most common cause of unexpected runtime, and is covered in Chapter 15.

1.3 Supported inputs and outputs

Category	Formats
`Layout in`	GDSII, OASIS, LEF/DEF
`Layout out`	GDSII, OASIS
`Netlist in`	SPICE, CDL, structural Verilog

Category	Formats
`Netlist out`	SPICE, CDL
`Rules`	SVRF, LRF
`Results`	Calibre-compatible ASCII RDB, CSV, binary results database, marker files
`Parasitics`	SPEF, DSPF, SDF

2 Installation

2.1 Supported operating systems

LVR binaries are built inside a CentOS 7 container and statically linked, so the resulting executable does not depend on host dynamic libraries. CentOS 7 provides glibc 2.17, which is present in or compatible with every enterprise Linux distribution released in the last decade. In practice this means LVR runs unmodified on essentially any Linux a semiconductor design environment is likely to use.

Distribution	Versions	Notes
`CentOS`	7	Build and validation baseline. glibc 2.17, kernel 3.10.x.
`Red Hat Enterprise Linux`	7.x, 8.x	RHEL 7 shares CentOS 7's lineage and runs natively. RHEL 8 uses a newer kernel and glibc; static linking makes this immaterial. Verified on both command-line and graphical server installations.
`Rocky Linux`	8, 9	Rocky 8 is binary-compatible with RHEL 8. Rocky 9 uses newer components; static linking keeps execution unaffected. Both GUI and headless operation supported.
`SUSE Linux Enterprise Server`	12, 15	SLES 12 tested for DRC and LVS flows. Static linking bypasses the SUSE libtool chain constraints.
`Ubuntu`	18.04 LTS, 20.04 LTS	Verified on Bionic (glibc 2.27) and Focal (glibc 2.31). GUI works under GNOME, KDE and Xfce.
`macOS`	11 and later	AMS Layout Editor and graphics view; Qt 6 with hardware acceleration.

2.2 Hardware

Resource	Minimum	Recommended
`Memory`	16 GB	64 GB for full-chip designs
`Cores`	1	16 or more; LVR scales close to linearly
`Disk`	2 GB for the installation	Additional space proportional to design size for results databases
`Display`	None required for headless operation	1920×1080 or better for the GUI

2.3 Running the installer

LVR ships with a graphical installer built on the Qt Installer Framework. Six steps, with progress shown in the navigation list at the left.

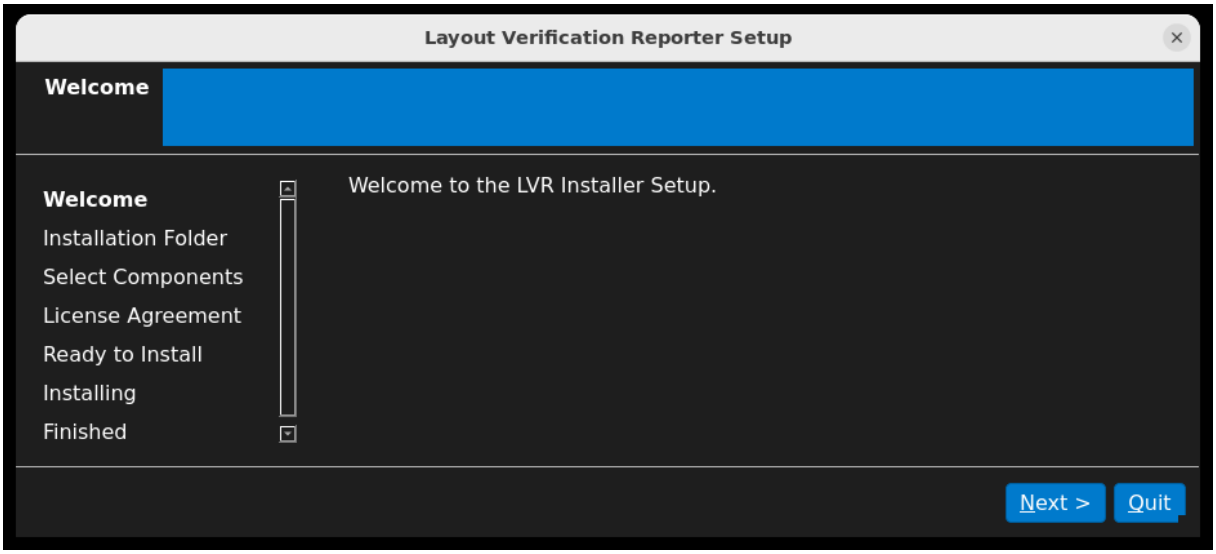

Figure 1. Installer welcome page.

Step	Page	Action
1	`Welcome`	Click Next.
2	`License Agreement`	Read and accept the terms.
3	`Installation Folder`	Select a writable path. The default is <code>/opt/lvr</code> or a directory under the user's home.
4	`Dependency Installation`	On Red Hat-based systems the installer runs <code>dnf</code> or <code>yum</code> to install the Qt runtime, Tcl and Perl. A progress window is shown while these complete.
5	`Select Components`	All components preselected. Click Next.
6	`Finished`	Click Finish.

2.4 After installation

The executable is installed at:

```
/opt/lvr/bin/LVR
```

Prepare the environment by sourcing the supplied script. Add this to the shell profile of any account that will run LVR:

```
source /opt/lvr/setup_env.sh
```

Confirm the installation:

```
% LVR
(the Tcl console opens; enter 'help' to list commands, then exit)
```

3 Your First Run

This chapter runs DRC on a small design end to end. It assumes a layout file, a rule file and a known top cell name.

3.1 Write the script

Create a file called first_run.cmd:

```
# initialize the tool database in DRC mode
init_tool_db DRC

# configure
set_technode sky130
set_top gcd1_sky130hd
set_num_cores 4
set_die_area 0 0 106060 106060

# read rules, then the design
read_lrf ../rule/sky130_DRC001.lrf
read_gds ../gds/gcd1_sky130hd.gds

# commit
load_db DRC

# run
no_compare 1
drc gcd1_sky130hd

# report and review
set_report_file_drc ./out/drc.rpt
report_drc
load_gui
```

3.2 Run it

```
% LVR first_run.cmd
```

LVR executes each command in order and opens the graphical interface when it reaches load_gui.

3.3 Review the results

In the Error Browser, work through the tabs in order:

Order	Tab	What to look for
1	`Errors and Warnings`	Fatal messages first. A missing file or an unmatched top cell will appear here and explains an empty result.
2	`<Engine> Summary Report`	Per-rule counts. Whether failures are one systematic problem or many independent ones.

Order	Tab	What to look for
3	`Histograms`	The spatial heat map. A hot region usually means one block or one repeated cell accounts for most of the count.
4	`Marker Browser`	Individual violations. Double-click a row to centre it in the layout.

3.4 Run it headless

For regression use, replace load_gui with three lines that suppress the interface and make output byte-comparable between runs:

```
cmd_line_only
set_testmode 1
disable_logger_timestamp
```

4 Core Concepts

4.1 Modes

A mode is selected by init_tool_db and determines which engine is armed, which rule constructs are honoured and which report writers are available. A command issued in a mode it does not apply to is accepted and ignored, so a shared setup file can safely contain commands for several modes.

Two modes can be used in one session: run the first, then set_tool_db to switch context to a second database without re-reading the layout.

4.2 The database lifecycle

The single most important structural rule in LVR is that load_db separates reading from running. Everything is read before it; everything is run after it.

Phase	Commands	Rule
Initialize	`init_tool_db`	First command. Nothing may precede it.
Configure	`set_technode, set_top, set_num_cores, set_die_area, set_halo_width`	Before any read, because parsing consults the layer map and the hierarchy root.
Read rules	`read_svrf, read_lrf`	Layer declarations before the rule files that reference the layers they declare.
Read design	`read_gds, read_spice, read_verilog, read_lef, read_def`	After rules.
Commit	`load_db, load_def_db`	Builds geometry, layer and hierarchy structures. Nothing may be read after this.
Run	`drc, lvs, erc, xor_check, density, pex`	Requires a committed database.
Report	`set_report_file_*, report_*, load_gui`	After the run.

4.3 Regions, cores and the halo

set_num_cores sizes the worker pool. set_num_regions sets how finely the die is partitioned. More regions than cores improves load balance on designs with uneven density, at the cost of more boundary work.

set_halo_width is the band each worker reads beyond its own region so that rules spanning a boundary are evaluated correctly. It must be at least the largest interaction distance in the rule deck. A halo that is too small produces missed violations at region boundaries — a failure mode that is easy to miss because the results look plausible. A halo that is too large simply costs redundant work.

4.4 Layers

Layer names come from the rule deck's declarations, or from lvr_layer and its companions in a technology file. Rule files refer to layers by name, so the declared names must match the foundry design rule manual.

lvr_kind classifies each layer as metal, cut, diffusion, poly, well, implant or text. The classification drives default behaviour in connectivity extraction, antenna accumulation and density measurement. Misclassifying a layer produces plausible but wrong results rather than an error message, so it is worth checking whenever numbers look inexplicable.

4.5 Results and waivers

Results are held in a results database and written out by the report_* commands. The DRC report format is Calibre-compatible ASCII RDB, so LVR output loads directly into existing marker viewers and correlation scripts.

A waiver suppresses a known, reviewed violation. LVR records each waiver against a SHA-256 hash of the geometry it was written for. If that geometry changes, the waiver stops applying automatically. This is what prevents a waiver written for one revision from silently hiding a new violation in the next — the failure mode that makes hand-maintained waiver lists dangerous.

5 Design Rule Checking

5.1 Minimal DRC script

```
init_tool_db DRC
set_technode sky130
set_top gcd1_sky130hd
read_lrf ../rule/sky130.lrf
read_gds ../gds/gcd1_sky130hd.gds
load_db DRC
drc gcd1_sky130hd
set_report_file_drc ./out/drc.rpt
report_drc
```

5.2 Running a production foundry deck

A foundry deck arrives as a top-level file that includes several dozen others. LVR does not follow those includes; the script issues one read_svrf per include file, in the order the deck's own top-level file lists them. Order matters because a rule file may reference a layer an earlier file derived.

```
init_tool_db DRC
set_technode sg13g2
set_top chip_top
set_num_cores 16
set_die_area -2000 0 2000 2000

# layer declarations first
read_svrf ../rule/foundry_deck/Include/tech.layers.svrf
# then derivations
read_svrf ../rule/foundry_deck/Include/tech_boolean.drc.svrf
read_svrf ../rule/foundry_deck/Include/tech_derived_layers.drc.svrf
# then the rule files, in deck order
read_svrf ../rule/foundry_deck/Include/tech_beol_m1.drc.svrf
read_svrf ../rule/foundry_deck/Include/tech_crackstop.drc.svrf
# ... one line per include file ...
read_svrf ../rule/foundry_deck/Include/tech_pad.drc.svrf

read_gds ../gds/chip_top.gds
load_db DRC
no_compare 1
drc chip_top
set_report_file_drc ./out/drc.rpt
report_drc
```

Comment out rather than delete the read_svrf lines for files not needed in a given run — a density deck during a geometry-only run, for example. The script then remains a faithful record of the deck it was built from, and re-enabling a section is a one-character edit.

5.3 Bounding a debug run

Against a deck under development, a single broken rule can produce millions of markers. drc_max_error caps the recorded count:

```
drc_max_error 100
```

A capped run reports a subset. Its counts must not be compared against an uncapped golden database — the cap must be lifted on both sides, or accounted for explicitly, before the numbers mean anything.

5.4 Reading DRC results

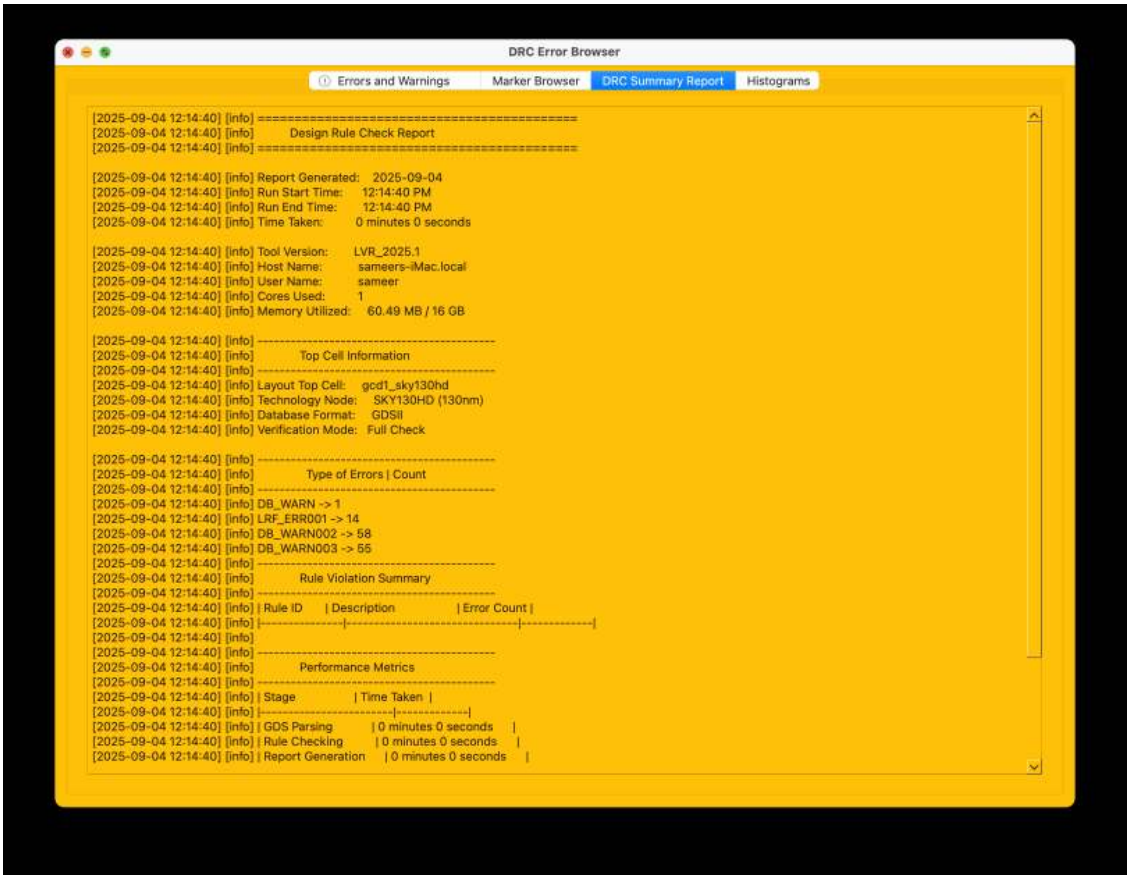

Figure 2. DRC Summary Report: report generation timestamp, tool version, layout and rule file names, technology node, hierarchy mode, per-rule error counts and stage timing.

6 Layout Versus Schematic

LVS extracts devices and connectivity from the layout, normalizes the source netlist, and compares the two graphs. Most LVS setup effort goes into three things: telling the extractor which devices exist, telling it which nets are supplies, and telling it how names on the two sides correspond.

6.1 Minimal LVS script

```
init_tool_db LVS
set_technode sg13g2
set_top DiffAmp_NOFILL

read_svrf ../rule/sg13g2.extract.cal
read_gds ../gds/DiffAmp_NOFILL.gds
set_lvs_spice_file ../netlist/DiffAmp.cdl

set_lvs_power_pin VDD
set_lvs_ground_pin VSS

set_lvs_use_svrf_device_rules 1
set_lvs_use_svrf_connect 1

load_db LVS
lvs DiffAmp_NOFILL
set_lvs_report_file ./out/lvs.rpt
report_lvs
```

6.2 Device recognition

For a foundry deck, always let the deck define the devices:

```
set_lvs_use_svrf_device_rules 1
set_perform_svrf_device_parse 1
```

Only the deck knows the process device definitions. The built-in defaults exist for quick checks against a simple PDK, not for sign-off.

Where devices are declared explicitly, enable exactly the families the technology uses. Enabling one the process does not have costs runtime; omitting one it does have produces a device count mismatch that looks like a layout error but is a setup error.

6.3 Supplies

Supply declarations affect short detection, ERC, antenna accumulation and bulk terminal resolution. Declare each domain separately rather than relying on name patterns:

```
set_lvs_power_pin VPWR
set_lvs_ground_pin VGND
set_lvs_power_pin1 VDDA
set_lvs_ground_pin1 VSSA
```

```
set_lvs_connect_power_global 1
```

For PDKs using four-terminal devices, map the bulk terminal names:

```
set_lvs_extract_bulk_terminal 1
set_lvs_alias_vpb VNW
set_lvs_alias_vnb VPW
```

6.4 Hierarchical versus flat

Setting	Use when	Trade-off
`set_lvs_hierarchical 1`	Layout and schematic hierarchies correspond	Faster; errors localized to the cell where they occur
`set_lvs_flatten_cells 1`	Hierarchies differ, for example after logic restructuring, or where the layout merges cells the netlist keeps separate	Always works; slower, and errors are reported at top level

Exclude cells that carry no useful connectivity:

```
set_lvs_skip_fill_cells 1
set_lvs_skip_empty_cells 1
```

6.5 Diagnosing a mismatch

Work in this order. Each step isolates a different stage.

#	Check	How	Means
1	Device counts	`set_lvs_report_device_counts 1`	A count mismatch is an extraction or netlist problem, not a matching problem. Stop here and fix it.
2	Extracted netlist	`lvs_write_layout_from_db ./out/extracted.cdl`	Shows what the comparison actually sees from the layout.
3	Normalized schematic	`lvs_write_schematic_from_spice in.cdl ./out/norm.cdl`	Diff against step 2 to see whether the difference originates in extraction or in the source netlist.
4	Shorts and opens	`set_lvs_check_shorts 1, lvs_check_open 1, set_lvs_open_report_path 1`	A short merges two nets and cascades into many apparent mismatches. Fix shorts before investigating matching.
5	Unmatched instances	`set_lvs_match_report_unmatched 1`	Names the instances that could not be paired.
6	Properties	`set_perform_lvs_property_check 1`	A topologically correct but mis-sized layout passes without this.

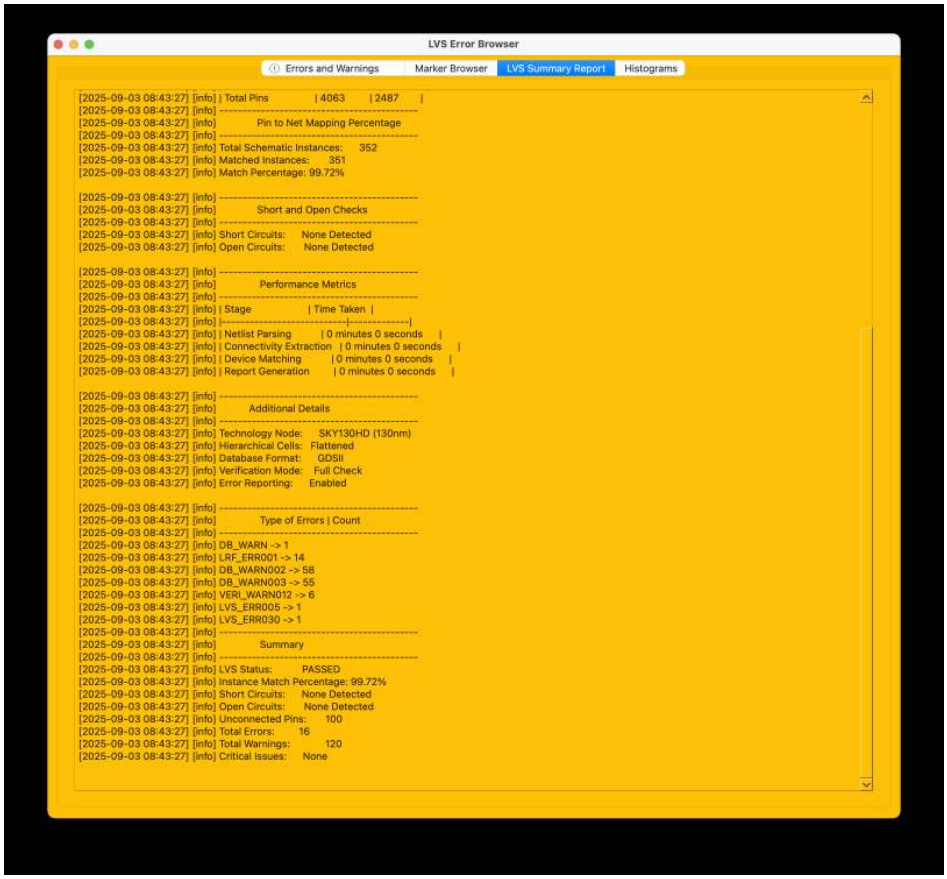

Figure 3. LVS Summary Report: short and open counts, device and instance match percentages, net and device totals, per-stage timing.

7 Electrical Rule Checking

ERC operates on extracted connectivity and reports conditions that are geometrically legal but electrically wrong. Because most ERC checks are defined relative to power and ground, supply declarations are a prerequisite rather than an option.

```
init_tool_db ERC
set_technode sky130
set_top gcd1_sky130hd
read_lrf ../rule/sky130.lrf
read_gds ../gds/gcd1_sky130hd.gds

set_lvs_power_pin VPWR
set_lvs_ground_pin VGND

set_erc_check_floating_gate 1
set_erc_check_floating_bulk 1
set_erc_check_short_to_power 1
set_erc_check_short_to_ground 1
set_erc_check_multi_driver 1
set_erc_check_undriven_power 1
set_erc_check_power_short 1

load_db ERC
erc gcd1_sky130hd
set_report_file_erc ./out/erc.rpt
report_erc
```

ERC can also run inside the LVS pipeline, reusing the connectivity already extracted rather than requiring a separate pass:

```
set_perform_lvs_erc 1
set_lvs_erc_report_file ./out/erc.rpt
```

In the graphics view, ERC results are best read in the connectivity view, where shapes are coloured by net rather than layer. A short between two nets then appears as a single connected region in a colour that should not exist.

Figure 4. Connectivity view during ERC review. Shapes coloured by net; the Selection Panel reports the nets and pins involved.

8 Layout Differencing

XOR reports every region present in one layout and absent from the other. It is the mechanism for ECO verification, revision sign-off, and confirming that a fill or metal-fill step changed only what it was supposed to.

```
init_tool_db XOR
set_technode sky130
set_top gcd1_sky130hd
set_xor1_gds ../gds/revA.gds
set_xor2_gds ../gds/revB.gds
load_db XOR
xor_check gcd1_sky130hd
set_gui_xor_result 1
set_report_file_xor ./out/xor.rpt
report_xor
load_gui
```

Engine	Precision	Use for
`run_xor / xor_check`	Exact difference geometry	ECO verification and revision sign-off, where the exact extent of every difference matters
`run_fastxor`	Whether and approximately where layouts differ	Screening across many cells. Follow up with the exact engine on any cell that reports a difference

set_gui_xor_result 1 loads difference regions into the graphics view as markers, so they can be navigated exactly like DRC violations.

9 Density and Manufacturability

Density checking measures the fill ratio of each named layer within a sliding window and reports windows outside the permitted range.

```
init_tool_db DENSITY
set_technode generic22
set_top analog_core
read_svrf ../rule/foundry_deck/density.cal
read_gds ../gds/analog_core.gds

density_layer Metal1
density_layer Metal2
density_layer Metal3
density_pixel 1000
density_window 100000 100000 50000 50000 1

load_db DENSITY
density analog_core
set_report_file_density ./out/density.rpt
report_density
```

9.1 Getting density to correlate

Density results are unusually sensitive to setup. Four things must match the foundry rule exactly or the numbers will not correlate with a reference tool, even when the geometry is identical:

#	Item	Why
1	`Window size`	Determines the measurement area.
2	`Step`	A step smaller than the window gives overlapping windows, which is what most foundry decks require. Non-overlapping scanning misses local excursions.
3	`Pixel size`	Must be small relative to the smallest feature contributing to density, or the ratio is quantized.
4	`Layer set`	Only layers named by <code>density_layer</code> are measured. A missing layer silently lowers the measured density.

A further correlation trap is input scope. Where a reference flow merges several GDS files before measuring and the LVR run reads only one, layers present only in the omitted files will be absent from LVR's result. This is an input difference, not a tool difference, and is worth eliminating before investigating anything else.

9.2 Antenna and DFM

```
init_tool_db ANTENNA
set_technode generic12
set_top top_soc
read_svrf ../rule/foundry_deck/tech_antenna.drc.svrf
read_gds ../gds/top_soc.gds
set_lvs_extract_diode 1
load_db ANTENNA
```

```
antenna top_soc
set_report_file_antenna ./out/antenna.rpt
report_antenna
```

Antenna checking accumulates conductor area connected to each gate at each metallization stage and flags nets whose ratio exceeds the process limit. Diode extraction must be enabled, or protection diodes will not be recognized and every protected net will report a false violation.

10 Parasitic Extraction

PEX converts layout geometry into an RC network. Resistance is modelled by ladder segmentation of each conductor; capacitance decomposes into ground, fringe and coupling components. IR drop and electromigration analysis run on the extracted network.

```
init_tool_db PEX
set_technode generic22
set_top top_soc
read_gds ../gds/top_soc.gds
set_pex_tech_file ../tech/tech.pextech

set_pex_run_extraction 1
set_pex_extract_resistance 1
set_pex_extract_via_resistance 1
set_pex_extract_capacitance 1
set_pex_extract_ground_cap 1
set_pex_extract_fringe_cap 1
set_pex_extract_coupling_cap 1

set_pex_ladder_seg_len_um 1.0
set_pex_max_coupling_dist_um 2.0
set_pex_inter_net_coupling_only 1
set_pex_skip_power_nets 1

set_pex_cap_unit FF
set_pex_res_unit OHM
set_pex_report_spef ./out/top_soc.spef

set_pex_num_threads 16
load_db PEX
pex top_soc
```

10.1 Controlling SPEF size

Setting	Effect
`set_pex_inter_net_coupling_only 1`	Discards intra-net coupling. Large size reduction with no loss for crosstalk analysis.
`set_pex_skip_power_nets 1`	Excludes supplies, which dominate the parasitic count. Do not use if IR drop analysis is wanted.
`set_pex_min_cap_ff / set_pex_min_coupling_cap_ff`	Discards negligible capacitors.
`set_pex_min_resistance_ohm`	Shorts negligible resistors.
`set_pex_rc_reduction 1`	Reduces the network before writing.
`set_pex_ladder_seg_len_um`	Longer segments mean fewer nodes and a coarser distributed-resistance model.

10.2 IR drop and electromigration

```
set_pex_run_ir_drop 1
set_pex_supply_voltage 0.8
set_pex_ir_drop_threshold_mv 40.0
set_pex_cg_max_iter 1000
set_pex_cg_tolerance 1e-6
set_pex_report_ir ./out/ir.rpt

set_pex_run_em 1
set_pex_em_violation_ratio 1.0
set_pex_report_em ./out/em.rpt
```

If a run reaches set_pex_cg_max_iter the solver has not converged and the IR results should not be trusted. Raise the iteration cap or loosen the tolerance, and check that the supply network is fully connected — an unconnected supply island is a common cause of non-convergence.

10.3 Correlating against a reference extractor

```
compare_spefs golden.spef ./out/top_soc.spef
```

Reports per-net differences in resistance and capacitance. Run this whenever an extraction setting changes, to quantify the effect rather than assume it.

11 LEF/DEF Flow

The LEF/DEF flow checks a placed-and-routed design against a set of individually armed checks rather than a rule deck. Every check defaults to disabled, so a run checks only what the script enables.

```
init_tool_db DRC
set_lefdef_flow 1
set_top design_top

read_lef ../lef/tech.lef
read_lef ../lef/stdcells.lef
read_def ../def/design.def
load_def_db

check_metal_spacing 1
check_cut_spacing 1
check_minimal_width 1
check_min_cut 1
check_cut_enclosure 1
check_shorts 1
check_open_check 1
check_off_grid 1
check_wrongway 1
check_antenna 1

run_lefdef_drc
set_report_file_lefdef_drc ./out/lefdef_drc.rpt
report_lefdef_drc
```

11.1 Narrowing a triage run

Command	Effect
`no_cells 1`	Skips standard-cell instances, isolating routing-layer violations from cell-internal ones.
`no_obs 1`	Skips LEF OBS obstructions, useful where a library's obstruction modelling is pessimistic enough to mask real violations.
`no_pwr 1`	Excludes power and ground nets. Supplies are large and highly connected, so this substantially shortens a signal-routing debug loop.
`no_vias 1`	Skips vias, separating metal-layer from cut-layer violations.

12 Migrating from an Existing Flow

Because LVR reads SVRF natively, migration is largely a matter of translating the run script rather than the rules.

12.1 What transfers unchanged

Item	Status
SVRF rule decks	Read directly. No translation.
GDSII and OASIS layouts	Read directly.
SPICE and CDL netlists	Read directly.
Layer maps	Read directly via <code>set_layer_map_file</code>.
ASCII RDB results	LVR writes Calibre-compatible ASCII RDB, so existing marker viewers and correlation scripts continue to work.
Deck includes	One <code>read_svrf</code> per include file, in deck order.

12.2 What needs rewriting

The run script. An existing runset is replaced by an LVR command script following the seven-phase structure of section 4.2. The rule reads are mechanical; the configuration and reporting sections need mapping.

12.3 Correlation checklist

Before concluding that a count difference between LVR and a reference tool is a tool difference, eliminate these. In practice most initial discrepancies are on this list.

#	Check	Symptom if wrong
1	Same input files, and the same number of them	Whole rules empty in one tool. Merging several GDS files versus reading one is the most common cause.
2	Same deck version and the same set of includes	Individual rules differ while everything else matches.
3	Marker caps lifted on both sides	Counts agree up to a round number such as 1000, then diverge.
4	Same die area	Geometry near the boundary missing from one side.
5	Halo at least the largest interaction distance	Sporadic missing violations at region boundaries.
6	Same top cell and the same hierarchy mode	Counts differ by a factor related to instance count.
7	Same layer map and datatypes	Whole layers empty, or geometry appearing on the wrong layer.
8	Same rounding convention on derived dimensions	Off-by-one-database-unit differences on marginal geometry.

Separate genuine mismatches from measurement artifacts before reporting a correlation figure. A capped rule, a deck-version difference and a missing input file are all measurement artifacts, and counting them as tool failures understates correlation.

13 Reports and Results

13.1 Report commands

Each engine has a matched pair: a set_report_file_* that names the path and a report_* that writes it. Both follow the run. If no report file is named, output goes to the console.

13.2 Output artifacts

Artifact	Command	Contents
Report	`set_report_file_drc`, `set_lvs_report_file`	Pass or fail status, counts, summary of every violation class
Marker file	`set_lvs_marker_file`	Layout coordinates of every violation
CSV	`set_lvs_csv_file`, `set_lvs_enable_csv_export`	Spreadsheet or script post-processing
Results database	`set_lvs_db_file`, `set_lvs_enable_db_export`	Full marker detail for later interactive review without re-running
Extracted netlist	`set_lvs_extracted_netlist`	The netlist the comparison saw from the layout
Signature	`set_lvs_signature_file`	Hash of every input, for a reproducible sign-off record
SPEF / DSPF / SDF	`set_pex_report_spef and companions`	Parasitics

13.3 Message severities

Severity	Meaning	Action
`Info`	Progress and configuration echoes	None
`Warning`	Did not stop the run but may affect results, for example a rule referencing an empty layer	Investigate before trusting the result
`Error`	A violation or a run-level failure; the run continues	Review in the Marker Browser
`Fatal`	The run cannot continue	Read this tab first when a run produces nothing

Error identifiers are structured by engine — DRC_ERR001, LVS017, DB ERR005 — and the Errors and Warnings tab groups by identifier. Grouping is what makes a run with thousands of markers reviewable, because the categories reveal at once whether the failures are one systematic problem or many independent ones.

14 Diagnostic Message Reference

LVR messages carry a structured identifier. The prefix names the subsystem that raised the message and the number identifies the condition. Identifiers are stable across releases, so they are safe to key automation on.

Prefix	Subsystem	Raised during
`DB ERRnnn`	Database and input handling	Reading and committing inputs, before any engine runs
`DRC ERRnnn`	Design rule check	Geometry checking
`LVS ERRnnn`	Layout versus schematic	Extraction and netlist comparison
`MLVS ERRnnn`	Macro LVS	Macro-level comparison
`ERC ERRnnn`	Electrical rule check	Connectivity rule evaluation
`SVS ERRnnn`	Schematic versus schematic	Netlist-to-netlist comparison
`XOR ERRnnn`	Layout difference	Geometric differencing

Braces {} in a message template are substituted at runtime with the relevant cell, net, layer or file name.

Read DB messages first. A database error means no engine ran at all, so every downstream symptom is a consequence rather than a cause. The Fatal tab of the Message Panel is the fastest route to these.

14.1 Database and Input Handling

Code	Message template	Meaning and impact
`DB ERR001`	`Top cell is not specified`	The top cell name required to anchor hierarchy was not specified. Modules cannot resolve a root for analysis; runs aborted.
`DB ERR004`	`Review and fix the DB loading errors and re-run LVR: DENSITY cannot proceed`	There were errors while loading inputs; modules depending on DB will not proceed. Subsequent runs (LVS/DRC/etc.) are blocked.
`DB ERR005`	`No DB instance found for {}, create a DB using init tool db command`	No active DB instance is present for the requested tool module. Module cannot access state or write results.
`DB ERR006`	`The file {} could not be opened`	An input file could not be opened for reading. Run configuration is incomplete; module may fail early.
`DB ERR007`	`LRF file not found`	LRF (layout rules file) not found where expected. Cannot run rule-driven checks.
`DB ERR008`	`Module-SPICE file not found`	Module SPICE file not found by the input forms/checks. Dependent flows (LVS/ERC/SVS) cannot proceed.
`DB ERR009`	`Macro CDL Library not found`	Macro CDL library not found. Macro expansion blocked in flows requiring CDL.
`DB ERR010`	`DB for {} already exists`	A DB with the requested name already exists. May cause accidental overwrite or stale results.
`DB ERR011`	`Summary Report File not found`	Summary report file not found when validating inputs. Report generation cannot proceed for that module.
`DB ERR012`	`Markers File not found`	Marker file not found while validating GUI/report options. Viewer cannot load markers; reporting degraded.

Code	Message template	Meaning and impact
DB ERR014	Macro Timeout is less than 1 second	Macro timeout parameter is too small. Macro extraction may not complete.
DB ERR015	Instance Matching Timeout is less than 60 seconds	Instance matching timeout is too small. Equivalence search may terminate early.
DB ERR016	Number of OpenMP threads is invalid	Number of threads/cores parameter is invalid (less than 1). Parallel engine disabled or errors out.
DB ERR017	One or more input fields (LineEdits) are missing.	One or more required UI input fields are empty. Cannot validate or start run.
DB ERR018	Number of Cores is more than the available cores for this machine	Requested core count exceeds available machine cores. Scheduling will fail or be clamped.
DB ERR021	Max error is less than 1	Max error count is set below 1. Error reporting suppressed unnaturally; runs may abort.
DB ERR022	Max error (option) is less than 1	Option max error limit is set below 1. Option-related diagnostics suppressed.

14.2 Design Rule Check

Code	Message template	Meaning and impact
DRC ERR009	Module {} not found in Layout DB for {}, DRC cannot proceed	The requested module (top cell) was not found for DRC context, usually due to mismatched naming or missing GDS. DRC cannot target the intended block; run aborted.
DRC ERR023	DRC has aleady been run, to re-run, create a fresh DB	A DRC run already exists in this DB; the tool enforces clean re-runs only. Stale markers/reports could confuse interpretation; run blocked.
DRC ERR024	DRC DB has not been set, please initialize DRC DB to run drc	DRC DB has not been initialized for the current session. Cannot load rule deck or write markers; DRC aborted.

14.3 Layout Versus Schematic

Code	Message template	Meaning and impact
LVS ERR003	Pin {} of instance {} is open and unconnected	A pin on a placed instance in the layout has no resolved connection in the extracted netlist. The schematic expects this pin to be tied to a net, but the layout leaves it floating. Connectivity mismatch leading to LVS failure; floating pins can cause func- tional indeterminism.
LVS ERR005	The instance {} of the schematic remained unmatched	Generic tool error. Operation failed or halted.
LVS ERR006	Macros cdl file not specified, LVS cannot proceed	LVS requires a macro CDL library so that black-box macros can be expanded into devices for equivalence checking. Macros compare as unknown; LVS cannot descend hierarchy and will halt or skip comparisons.
LVS ERR007	Module Spice file not specified, LVS cannot proceed	The primary schematic netlist (SPICE) is not provided for LVS. No reference graph to compare against; LVS cannot proceed.

Code	Message template	Meaning and impact
LVS ERR009	Module {} not found in Layout DB for {}, LVS cannot proceed	The named module (top cell) cannot be found in the loaded layout database. Stop condition for LVS or module-level checks; top-of-hierarchy cannot be re- solved.
LVS ERR017	Found short {} interacting with {} on layer {}	Two nets that should be isolated are found shorted in the layout extraction. Hard LVS mismatch; can imply catastrophic functional failure or reliability is- sues.
LVS ERR018	Open detected! Net {} is broken into {} disconnected parts	A single logical net is fragmented into multiple disconnected islands in the layout. Open circuit; signals cannot propagate; LVS mismatch.
LVS ERR019	The pin {} of instance of the layout {} is unconnected, this will lead to LVS mismatches	A pin of a layout instance is unconnected; typically reported with instance and pin names. Instance-level mismatch and potential functional failure at block pins.
LVS ERR020	Mismatch in pins and ports size for instance {}	The count of pins/ports between compared entities differs (e.g., layout vs netlist or left vs right schematic). Graph isomorphism cannot be established; LVS comparison terminates with mismatch.
LVS ERR023	LVS has aleady been run, to re-run, create a fresh DB	LVS is being invoked on a database where LVS results already exist, and the flow requires a fresh DB for a clean run. Stale artifacts may contaminate results; tool refuses to run.
LVS ERR024	LVS DB has not been set, please initialize LVS DB to run lvs	The LVS module's database context is not initialized. LVS cannot load inputs or write outputs; run aborted.
LVS ERR025	Required pin {} of instance {} is unconnected	A required pin reported by the LVS engine is unconnected, indicating incom- plete connectivity for the instance. Mismatch at the device interface; LVS equivalence fails.
LVS ERR026	Required pin {} of instance {} is unconnected	A required pin reported by the LVS engine is unconnected, indicating incom- plete connectivity for the instance. Mismatch at the device interface; LVS equivalence fails.
LVS ERR027	Required pin {} of instance {} is unconnected	A required pin reported by the LVS engine is unconnected, indicating incom- plete connectivity for the instance. Mismatch at the device interface; LVS equivalence fails.
LVS ERR028	Required pin {} of instance {} is unconnected	A required pin reported by the LVS engine is unconnected, indicating incom- plete connectivity for the instance. Mismatch at the device interface; LVS equivalence fails.

14.4 Macro LVS

Code	Message template	Meaning and impact
MLVS ERR002	Mismatch for the Gate Pin of macro {}	Macro LVS pin mismatch at specific terminals (e.g., Gate, Source). The pin type does not align between macro representation and extracted view. Macro equivalence fails; downstream LVS may cascade mismatches.
MLVS ERR003	Mismatch for the Source Pin of macro {}	Macro LVS pin mismatch at specific terminals (e.g., Gate, Source). The pin type does not align between macro representation and extracted view. Macro equivalence fails; downstream LVS may cascade mismatches.
MLVS ERR006	Macros cdl file not specified, MLVS cannot proceed	Macro CDL file not specified for MLVS run. Cannot expand macro internals; MLVS aborted.

Code	Message template	Meaning and impact
MLVS ERR007	Module Spice file not specified, MLVS cannot proceed	Macro SPICE file not specified for MLVS run. No reference netlist to compare; MLVS cannot proceed.
MLVS ERR009	Module {} not found in Layout DB for {}, MLVS cannot proceed	Macro top module not found in layout DB. MLVS cannot correlate macro instance.
MLVS ERR023	MLVS has aleady been run, to re-run, create a fresh DB	MLVS was already run in this DB; clean re-run required. Stale markers confuse results.
MLVS ERR024	MLVS DB has not been set, please initialize MLVS DB to run mlvs	MLVS DB not set for current session. Run cannot start.

14.5 Electrical Rule Check

Code	Message template	Meaning and impact
ERC ERR001	Pin {} of instance {} is shorted with power	A functional pin is accidentally tied to the power rail. Overdrive, leakage, or permanent assertion; can damage device.
ERC ERR002	Pin {} of instance {} is shorted with ground	A functional pin is accidentally tied to ground. Signal stuck-low; dynamic paths disabled; possible damage.
ERC ERR003	Input pin {} of instance {} is unconnected	An input pin is left unconnected in the schematic or extracted layout. Floating input causes unpredictable logic levels and current draw.
ERC ERR004	Output pin {} of instance {} is dangling and unconnected	An output pin has no load or is not connected onward. Dangling output; potential power waste or missing function.
ERC ERR005	Pin {} of instance {} is shorted to ground	A functional pin is shorted to ground (duplicate in messages may reflect detection at different stages). Permanent low; possible device stress.
ERC ERR006	Pin {} of instance {} is shorted to power	A functional pin is shorted to power. Permanent high; crowbar risk with downstream logic.
ERC ERR007	Net {} has multiple output drivers: {} and {}	A single net is driven by more than one output driver. Bus contention; reliability and timing hazards.
ERC ERR008	Power pin {} of instance {} is undriven	A power pin is not driven by any valid supply source. Undriven power pin makes cell behavior undefined.
ERC ERR009	Ground pin {} of instance {} is undriven	A ground pin is not connected to a ground network. Reference potential undefined; ESD and noise issues.
ERC ERR017	Found short {} interacting with {} on layer {}	A short is detected between two nets on a specific layer context. Electrical short; can cause functional failure.
ERC ERR019	The pin {} of instance of the layout {} is unconnected, this will lead to ERC mismatches	A layout instance exposes a pin that is not connected to any net, flagged at the ERC stage. Floating node; may cause leakage or undefined behavior.
ERC ERR020	Mismatch in pins and ports size for instance {}	Generic tool error. Operation failed or halted.
ERC ERR023	ERC has aleady been run, to re-run, create a fresh DB	Generic tool error. Operation failed or halted.

Code	Message template	Meaning and impact
ERC ERR024	ERC DB has not been set, please initialize ERC DB to run erc	Generic tool error. Operation failed or halted.

14.6 Schematic Versus Schematic

Code	Message template	Meaning and impact
SVS ERR001	No perfect left-side matching found!	No perfect matching found on left/right side during bipartite matching of instances. SVS cannot produce a 1:1 correspondence.
SVS ERR002	No perfect right-side matching found!	No perfect matching found on left/right side during bipartite matching of instances. SVS cannot produce a 1:1 correspondence.
SVS ERR005	The instance {} of the schematic remained unmatched	General SVS configuration or matching error. SVS halted or mismatch flagged.
SVS ERR006	Macros cdl file not specified, SVs cannot proceed	Macro CDL library required by SVS is missing. SVS cannot resolve macro pin/device details.
SVS ERR009	Module {} not found in Layout DB for {}, SVS cannot proceed	Top module not found in layout DB context when preparing SVS comparison. SVS cannot correlate hierarchy to schematic views.
SVS ERR010	Module Verilog file not specified, svs cannot proceed	Verilog files required for one/both SVS sides are not specified. SVS cannot build comparison graphs.
SVS ERR011	Module Verilog file not specified, svs cannot proceed	Verilog files required for one/both SVS sides are not specified. SVS cannot build comparison graphs.
SVS ERR012	Module Spice file not specified, svs cannot proceed	SPICE file(s) required by SVS are not specified. SVS cannot run in SPICE-based path.
SVS ERR013	Module Spice file not specified, svs cannot proceed	SPICE file(s) required by SVS are not specified. SVS cannot run in SPICE-based path.
SVS ERR020	Mismatch in pins and ports size for instance {}	Instance-level mismatch in the number of pins or ports between compared schematics. Graph matching fails; SVS not equivalent.
SVS ERR023	SVS has aleady been run, to re-run, create a fresh DB	SVS already run in this DB; requires fresh DB for clean re-run. Stale artifacts may mislead analysis.
SVS ERR024	SVS DB has not been set, please initialize SVS DB to run svs	SVS DB not initialized. Run cannot proceed.
SVS ERR031	Mismatch in the number of instances , Left = {} , Right= {}	Structural mismatch: instance counts, pin counts, port counts, or pin-to-net mapping differ between schematics. SVS comparison fails; designs are not equivalent.
SVS ERR032	Mismatch in the number of pins for the instance {}	Structural mismatch: instance counts, pin counts, port counts, or pin-to-net mapping differ between schematics. SVS comparison fails; designs are not equivalent.

Code	Message template	Meaning and impact
SVS ERR033	Mismatch in the number of ports for the instance {}	Structural mismatch: instance counts, pin counts, port counts, or pin-to-net mapping differ between schematics. SVS comparison fails; designs are not equivalent.
SVS ERR034	Mismatch in the number of pin to net matching for the instance {}	Structural mismatch: instance counts, pin counts, port counts, or pin-to-net mapping differ between schematics. SVS comparison fails; designs are not equivalent.

14.7 Layout Difference

Code	Message template	Meaning and impact
XOR ERR001	XOR Mismatch is found for a polygon on the layer {}	Geometric XOR found a mismatch polygon on a specific layer. Masks differ between reference and test; fabrication risk if unintended.
XOR ERR009	Module {} not found in Layout DB for {}, XOR cannot proceed	The specified top module was not found in the layout DB while preparing XOR inputs. XOR cannot align hierarchies; run blocked.
XOR ERR023	XOR has aleady been run, to re-run, create a fresh DB	An XOR run already exists in this DB; clean DB required for re-run. Old markers would pollute the result; run blocked.
XOR ERR024	XOR DB has not been set, please initialize XOR DB to run xor check	XOR DB is not initialized. Run cannot start.

15 Performance Tuning

15.1 Where runtime goes

Enable stage timing and read the Summary Report before changing anything. Tuning the wrong stage is the most common way to spend effort without changing the result.

```
set_lvs_report_timing 1
enable_logger_timestamp
```

15.2 Geometry tier

This is the largest single lever. LVR represents geometry at the lowest tier the design requires:

Tier	Handles	Relative cost
`Manhattan`	Axis-aligned rectilinear geometry only	Lowest
`45-degree`	Adds 45-degree edges, such as crackstop structures	Moderate
`General`	Arbitrary angles	Highest

A design containing a small amount of 45-degree geometry does not need the general tier. Moving a rectilinear design to a higher tier than it requires can cost an order of magnitude in runtime for no change in results, and the symptom — a run that used to take minutes now taking hours, with identical output — is distinctive enough to recognise.

15.3 Parallelism

Symptom	Likely cause	Action
Runtime does not improve with more cores	Too few regions; workers idle	`Raise <code>set_num_regions</code> above the core count`
One region dominates the run	Uneven density; one block far denser than the rest	`Raise the region count further to subdivide the dense area`
Missing violations near region boundaries	Halo smaller than the largest interaction distance	`Raise <code>set_halo_width</code>`
Excessive memory	Too many regions, or caching too aggressive	`Lower the region count; set <code>set_lvs_memory_limit_mb</code>`

15.4 LVS-specific

```
set_lvs_num_threads 16
set_lvs_use_rtree 1
set_lvs_short_use_rtree 1
set_lvs_use_cache 1
set_lvs_cache_size_mb 4096
set_lvs_hierarchical 1
set_lvs_skip_fill_cells 1
set_lvs_skip_empty_cells 1
```

The R-tree index and the extraction cache are the two settings that matter most on a design dominated by repeated standard cells. Hierarchical mode avoids re-extracting an identical cell for every placement.

15.5 Instrumentation cost

Diagnostic settings are not free. Verbose logging, per-node IR output, per-segment EM output and raw RC dumps all add work proportional to design size. Turn them on to diagnose a specific problem and off again afterwards; leaving them enabled in a production script is a quiet and persistent runtime cost.

16 Troubleshooting

Symptom	Most likely cause	Resolution
Run completes with zero markers on a design known to have violations	Top cell name does not match the layout, or the die area excludes the geometry	Confirm <code>set_top</code> against the GDS structure names, including case. Check <code>set_die_area</code> covers the full extent.
Whole rules report nothing	A layer the rule references is empty	Check the layer map and the datatypes. Confirm the layer declaration file was read before the rules that use it.
Run fails immediately with no report	Fatal condition during read	Read the Fatal tab of the Message Panel. Usually a missing input file or an unparseable rule file.
Report file is empty or absent after a run	The run did not reach the reporting stage	Check for Fatal messages. Confirm the <code>report_*</code> command follows the run command.
Counts agree with a reference tool up to a round number then diverge	A marker cap is active on one side	Lift <code>drc_max_error</code> or the equivalent on both sides.
Sporadic missing violations near region boundaries	Halo smaller than the largest rule interaction distance	Raise <code>set_halo_width</code>.
Runtime increased by an order of magnitude with no change in results	Geometry moved to a higher representation tier than the design requires	See section 15.2.
LVS device counts do not match	A device family is not being extracted	Enable <code>set_lvs_use_svrf_device_rules</code> so the deck defines devices. Check each <code>set_lvs_extract_*</code> family the technology uses.
Very large numbers of LVS mismatches	One or more shorts merging nets	Fix shorts first. A single short cascades into many apparent mismatches.
Every protected net reports an antenna violation	Diode extraction disabled, so protection diodes are not recognized	Set <code>set_lvs_extract_diode 1</code>.
Bulk connectivity errors go undetected	Bulk terminal not extracted	Set <code>set_lvs_extract_bulk_terminal 1</code> and map the terminal names.
IR drop results implausible	Conjugate gradient solver did not converge	Check whether the iteration cap was reached. Verify the supply network is fully connected.
Density does not correlate	Window, step, pixel or layer set differs from the rule	See section 9.1. Also confirm both flows read the same set of input files.
Sub-blocks appear rotated or mirrored	Orientation composition convention differs from the writing tool	Adjust <code>cell_orient_scheme</code>.
Results differ between two runs of the same script	The script or an input changed between runs	Enable <code>set_testmode 1</code> and <code>disable_logger_timestamp</code> so outputs are byte-comparable, and confirm the inputs with <code>set_lvs_signature_file</code>.
Waiver no longer suppresses a violation it used to	The underlying geometry changed, so the content hash no longer matches	Intended behaviour. Review the violation and reissue the waiver if still appropriate.

Appendix A Command Quick Reference

The commands needed most often. The Command Reference documents all 441.

Task	Command
Initialize in a mode	`init_tool_db DRC`
Set technology	`set_technode sky130`
Set top cell	`set_top my_top`
Set worker pool size	`set_num_cores 16`
Set die area	`set_die_area 0 0 100000 100000`
Set boundary halo	`set_halo_width 5000`
Read an SVRF rule file	`read_svrf ../rule/deck.cal`
Read an LRF rule file	`read_lrf ../rule/deck.lrf`
Read a layout	`read_gds ../gds/design.gds`
Read a netlist	`set_lvs_spice_file ../netlist/design.cdl`
Declare supplies	`set_lvs_power_pin VDD / set_lvs_ground_pin VSS`
Commit the database	`load_db DRC`
Suppress golden comparison	`no_compare 1`
Cap markers for a debug run	`drc_max_error 100`
Run DRC	`drc my_top`
Run LVS	`lvs my_top`
Run ERC	`erc my_top`
Run XOR	`xor_check my_top`
Run density	`density my_top`
Run extraction	`pex my_top`
Name a report	`set_report_file_drc ./out/drc.rpt`
Write a report	`report_drc`
Open the interface	`load_gui`
Force headless	`cmd_line_only`
Make output diffable	`set_testmode 1 / disable_logger_timestamp`
Write the extracted netlist	`lvs_write_layout_from_db ./out/extracted.cdl`
Compare two SPEF files	`compare_spefs golden.spef lvr.spef`

Appendix B Glossary

Term	Meaning
CDL	Circuit Description Language. A SPICE dialect used for schematic netlists.
DBU	Database unit. The integer grid layout coordinates are stored on.
DFM	Design for manufacturability. Advisory rules that improve yield.
DRC	Design rule check. Geometric verification against foundry rules.
DSPF	Detailed Standard Parasitic Format. Parasitics as a SPICE subcircuit.
EOL	End of line. The terminating edge of a routing segment, subject to stricter spacing rules than a general edge.
ERC	Electrical rule check. Verification of electrical conditions on extracted connectivity.
Halo	The band each worker reads beyond its own region so that rules crossing a region boundary are evaluated correctly.
LEF / DEF	Library Exchange Format and Design Exchange Format. Abstract cell library and placed-and-routed design.
LRF	Layout Rules File. LVR's native rule format.
LVS	Layout versus schematic. Comparison of an extracted layout netlist against a source netlist.
Marker	A recorded violation with its layout coordinates.
MLVS	Macro LVS. Comparison treating hard macros as pin-interface black boxes.
OASIS	Open Artwork System Interchange Standard. A layout format more compact than GDSII.
PEX	Parasitic extraction.
PRL	Parallel run length. Spacing that increases with the length over which two edges run parallel.
RDB	Results database. The Calibre-compatible ASCII marker format LVR writes.
Region	A partition of the die assigned to one worker thread.
SDF	Standard Delay Format. Back-annotated timing.
SPEF	Standard Parasitic Exchange Format.
SVRF	Standard Verification Rule Format. The rule language LVR reads natively.
SVS	Schematic versus schematic. Comparison of two netlists with no layout.
Waiver	A reviewed suppression of a known violation, recorded against a content hash of the geometry it applies to.
XOR	Geometric difference between two layouts.