LVR Application Notes

Layout Verification Reporter

Ten worked procedures for production verification flows

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. Ten application notes covering deck migration, geometry tiers, density correlation, LVS bring-up, waivers, runtime tuning, marker triage, Virtuoso integration, AMS antenna and ERC, and ECO XOR.

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

Contents

0 Three Ways to Run LVR

Every note in this document assumes one of three invocation styles. This chapter establishes them once.

0.1 Method 1 — batch script

The most direct and the only one suitable for automation. Write a command script, run it, review the results.

```
% LVR command_script.cmd
```

LVR executes each command in order. If the script ends in load_gui, the interface opens on the completed results; if it contains cmd_line_only, the run is fully headless and requires no display. This method suits batch farms, regression suites and CI pipelines.

0.2 Method 2 — interactive, script sourced

Launch LVR with no arguments and source the script from the Tcl panel:

```
% LVR

# then, in the Tcl panel entry field:
source ./cmd_files/gcd1.cmd
```

The interpreter persists after the script finishes, so settings can be adjusted and individual stages re-run without restarting. This is the right method while a script is still being developed.

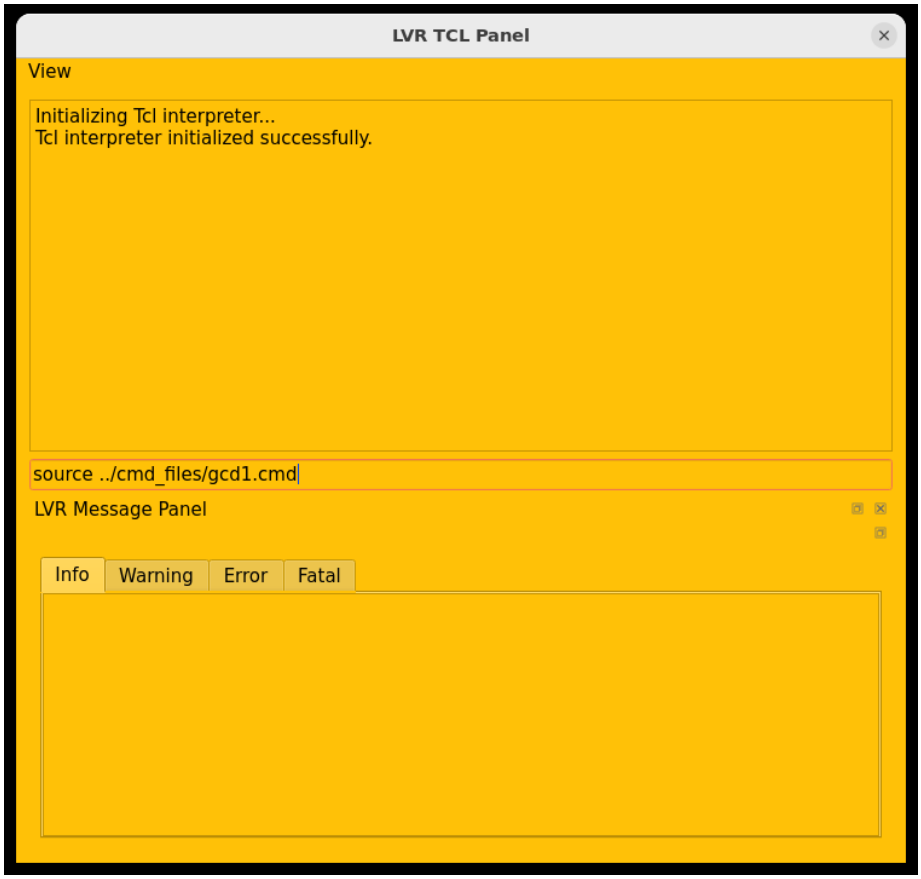

Figure 1. Tcl Panel after sourcing a script. The Message Panel below filters output by severity.

0.3 Method 3 — GUI configuration

Launch LVR with no arguments and define the run through the Configuration dialog. Fill in the Run Settings, Rules, Input, Output and Options tabs, then click Submit. Suitable for first-time use and for one-off runs; not reproducible without also saving the equivalent script.

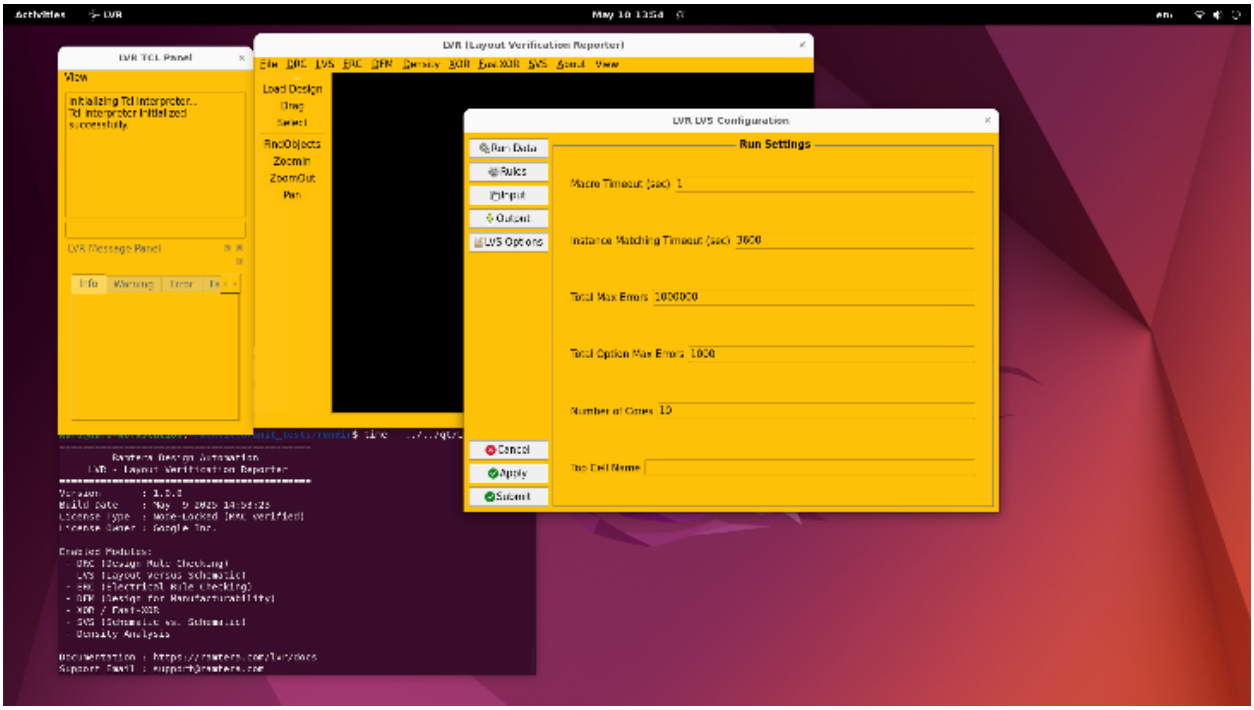

Figure 2. Configuration dialog, Run Settings tab.

AN-01 Migrating an SVRF Deck from an Existing Sign-off Tool

Problem	Applies to
A production SVRF deck is already qualified in another tool. It must run in LVR without being rewritten, and the results must be shown to correlate.	DRC, LVS, density. Any foundry node.

LVR reads SVRF natively, so the rules themselves do not change. What changes is the run script. Work in four stages.

Stage 1 — inventory the deck

A foundry deck is a top-level file that includes several dozen others. LVR does not follow those includes; the script issues one read_svrf per file. Extract the include list from the deck's top-level file, preserving order, and turn each into a line.

```
# from the deck's top-level file, in its own order:
read_svrf ../rule/foundry_deck/Include/tech.layers.svrf
read_svrf ../rule/foundry_deck/Include/tech_boolean.drc.svrf
read_svrf ../rule/foundry_deck/Include/tech_derived_layers.drc.svrf
read_svrf ../rule/foundry_deck/Include/tech_beol_m1.drc.svrf
# ... one line per include ...
read_svrf ../rule/foundry_deck/Include/tech_pad.drc.svrf
```

Order is not cosmetic. Layer declaration files must precede the rule files that reference the layers they declare, and derivation files must precede rules that use the derived layers. Keeping the deck's own order is the safest approach because the deck already satisfies these dependencies.

Stage 2 — build the run script

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

# ... the read_svrf block from stage 1 ...

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 reads for files not needed in a given run — a density deck during a geometry-only run, for example. The script then stays a faithful record of the deck it came from, and re-enabling a section is a one-character edit.

Stage 3 — validate before correlating

```
set_perform_lrf_check 1
```

This parses and validates the whole deck before the run begins, so a problem in the last include file is reported in seconds rather than after the geometry engine has already been running for an hour.

Stage 4 — correlate

LVR writes Calibre-compatible ASCII RDB, so existing marker viewers and correlation scripts work unchanged. Before treating any difference as a tool difference, eliminate these. Most first-pass discrepancies are on this list.

#	Check	Symptom when wrong
1	`Both flows read the same input files, and the same number of them`	Whole rules empty on one side. Merging several GDS files versus reading one is the most common cause of this.
2	`Same deck version, same include set`	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 on one side
5	`Halo at least the largest interaction distance in the deck`	Sporadic missing violations at region boundaries
6	`Same top cell, same hierarchy mode`	Counts differ by a factor related to instance count
7	`Same layer map and datatypes`	Whole layers empty, or geometry on the wrong layer
8	`Same rounding convention on derived dimensions`	Off-by-one-database-unit differences on marginal geometry

Report genuine mismatches separately from measurement artifacts. A capped rule, a deck-version difference and a missing input file are all artifacts; counting them as tool failures understates correlation and sends debug effort in the wrong direction.

AN-02 Non-Manhattan Geometry and Representation Tiers

Problem	Applies to
A design containing 45-degree structures such as crackstops runs far slower than an equivalent rectilinear design, with identical results.	DRC, XOR, density. Advanced nodes with crackstop, seal ring or analog 45-degree routing.

LVR represents geometry at the lowest tier a design requires. The tiers differ substantially in cost.

Tier	Handles	Relative cost	Typical use
`Manhattan`	Axis-aligned rectilinear geometry only	Lowest	Digital blocks, standard-cell arrays, most memory
`45-degree`	Adds 45-degree edges	Moderate	Crackstop and seal ring structures, some analog routing
`General`	Arbitrary angles	Highest	Curved or arbitrary-angle geometry, some RF structures

Recognising the symptom

The signature is distinctive: a run that previously took minutes takes hours, and the results are unchanged. Because output is identical, the problem is easy to attribute to something else. If runtime changed by an order of magnitude after a geometry or setup change and no marker count moved, the representation tier is the first thing to check.

Choosing the right tier

The mistake to avoid is jumping straight to the general tier because a design contains some non-rectilinear geometry. A crackstop drawn at 45 degrees needs the 45-degree tier, not the general one. The general tier exists for arbitrary angles, and using it where the 45-degree tier would suffice buys nothing and costs a great deal.

Design contains	Correct tier
Only axis-aligned rectangles and rectilinear polygons	`Manhattan`
45-degree crackstop, seal ring or chamfered corners	`45-degree`
Genuinely arbitrary angles, curves approximated by many short edges	`General`

Related pitfalls

Pitfall	Effect	Avoidance
Sizing operations on non-rectilinear shapes	Corner treatment differs between tiers, so a resize can chamfer corners that were expected to stay sharp	Verify sized output visually on a representative structure before trusting a whole-chip result
Angle and distance conventions in abutment rules	A rule expressed for rectilinear geometry may not mean what it appears to mean at 45 degrees	Check abutment and adjacency rules explicitly against a 45-degree test structure
Expensive diagnostics inside inner loops	Evaluating a size or count for a log message costs the same as computing it, even when the message is never printed	Keep verbose logging off in production runs

AN-03 Density Verification and Multi-File Input

Problem	Applies to
Density results do not correlate with a reference tool even though the geometry is identical.	DENSITY, DFM. Any node with metal or diffusion fill requirements.

Density is more sensitive to setup than any other check, because the measurement itself is defined by parameters the script supplies rather than by the geometry alone.

The four measurement parameters

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

#	Parameter	Failure mode when wrong
1	`Window size`	Every measured ratio differs; the error is systematic rather than localized
2	`Step`	A step equal to the window gives non-overlapping tiles, which miss local density excursions that overlapping windows catch. Most foundry decks require overlap.
3	`Pixel size`	If the pixel is not small relative to the smallest contributing feature, ratios are quantized and cluster on a few values
4	`Layer set`	Only layers named by <code>density_layer</code> are measured. A missing layer silently lowers measured density with no warning.

Input scope: the most common correlation trap

Where a reference flow merges several layout files before measuring and the LVR run reads only one, any layer present only in the omitted files will be absent from LVR's result. Whole rules then report nothing, which looks like a tool failure and is not.

This is an input difference. Establish that both flows see the same set of files before investigating anything else. The diagnostic is quick: if the rules that report nothing are exactly the rules covering layers carried in the files LVR did not read, the cause is confirmed.

Interpreting the result

When reporting density correlation, separate the categories. They call for different responses:

Category	Meaning	Response
`Genuine mismatch`	Both tools measured the same thing and disagreed	Investigate. This is the only category that indicates a tool problem.
`Input scope difference`	One tool saw layers the other did not	Align the inputs and re-measure.
`Deck version difference`	The two flows used different rule versions	Align the decks. Exclude from the correlation figure until aligned.
`Cap artifact`	One side stopped recording at a marker limit	Lift the cap on both sides and re-measure.

Category	Meaning	Response
`Derivation difference`	A derived layer was constructed differently	Compare the derivation chain rule by rule.

A correlation figure that lumps these together is not actionable. Reporting counts and per-rule status with the concrete cause attached to each exclusion is what lets someone else act on the result.

AN-04 Setting Up LVS on a New Technology

Problem	Applies to
LVS reports large device-count or connectivity mismatches on a design believed correct.	LVS, MLVS. Any new PDK or first-time technology bring-up.

Almost every first-run LVS failure on a new technology is a setup problem rather than a layout problem. Work through these in order; each isolates a different stage.

Step 1 — let the deck define the devices

```
set_lvs_use_svrf_device_rules 1
set_perform_svrf_device_parse 1
```

Only the foundry deck knows the process device definitions. The built-in recognition defaults exist for quick checks against simple PDKs, not for sign-off. Enabling a device family the technology does not use costs runtime; omitting one it does use produces a device count mismatch that looks like a layout error.

Step 2 — declare supplies and bulk terminals

```
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

# four-terminal PDKs
set_lvs_extract_bulk_terminal 1
set_lvs_alias_vpb VNW
set_lvs_alias_vnb VPW
```

Without bulk terminal extraction, bulk connectivity errors go undetected entirely — the comparison passes on a layout that has them. Declare each supply domain explicitly rather than relying on name patterns.

Step 3 — reconcile device multipliers

A schematic device carrying a multiplier corresponds to several physical devices in the layout. Unless multipliers are expanded, counts will not match even when the design is correct. Expansion happens during load_db. Confirm it worked by comparing the two netlists directly:

```
lvs_write_layout_from_db ./out/layout_extracted.cdl
lvs_write_schematic_from_spice ../netlist/design.cdl ./out/schematic_norm.cdl
```

Diffing these two files is the single most useful diagnostic in LVS. It separates an extraction problem from a source-netlist problem, and every later step is easier once that distinction is settled.

Step 4 — reconcile parallel and series structures

```
set_perform_lvs_parallel_reduce 1
set_perform_lvs_series_reduce 1
```

A layout drawn as several parallel fingers should match a schematic drawn as one wide device. Without reduction, it will not.

Step 5 — capture pins

```
set_lvs_use_pins_for_nets 1
set_lvs_use_svrf_connect 1
set_lvs_assign_internal_nets 1
set_lvs_internal_net_prefix LVR_N
```

Where terminals are marked with pin shapes rather than text labels, pin capture is what associates a net name with the geometry forming that terminal. Without it, nets come through unnamed and produce a large crop of spurious mismatches. Choose an internal net prefix that cannot collide with real net names.

Step 6 — set matching tolerances

```
set_lvs_match_symmetric_sd 1
set_perform_lvs_property_check 1
set_lvs_w_tolerance 0.01
set_lvs_l_tolerance 0.01
set_lvs_fin_pitch 0.048 # FinFET nodes only
```

Symmetric source/drain handling removes a whole class of spurious mismatches arising purely from terminal ordering. Property checking is what catches a topologically correct but mis-sized layout; a small tolerance absorbs rounding differences without masking real sizing errors. On FinFET nodes, an incorrect fin pitch makes every width comparison wrong by a constant factor.

Step 7 — read the result in the right order

#	Look at	If it is wrong
1	`Device counts by type`	Stop. This is extraction, not matching. Return to steps 1 and 3.
2	`Shorts`	Fix these before anything else. One short merges two nets and cascades into many apparent mismatches.
3	`Opens`	Enable <code>set_lvs_open_report_path 1</code> to localize the break rather than only naming the net.
4	`Unmatched instances`	Enable <code>set_lvs_match_report_unmatched 1</code> to name them.
5	`Property mismatches`	Sizing. Check tolerances and fin pitch.

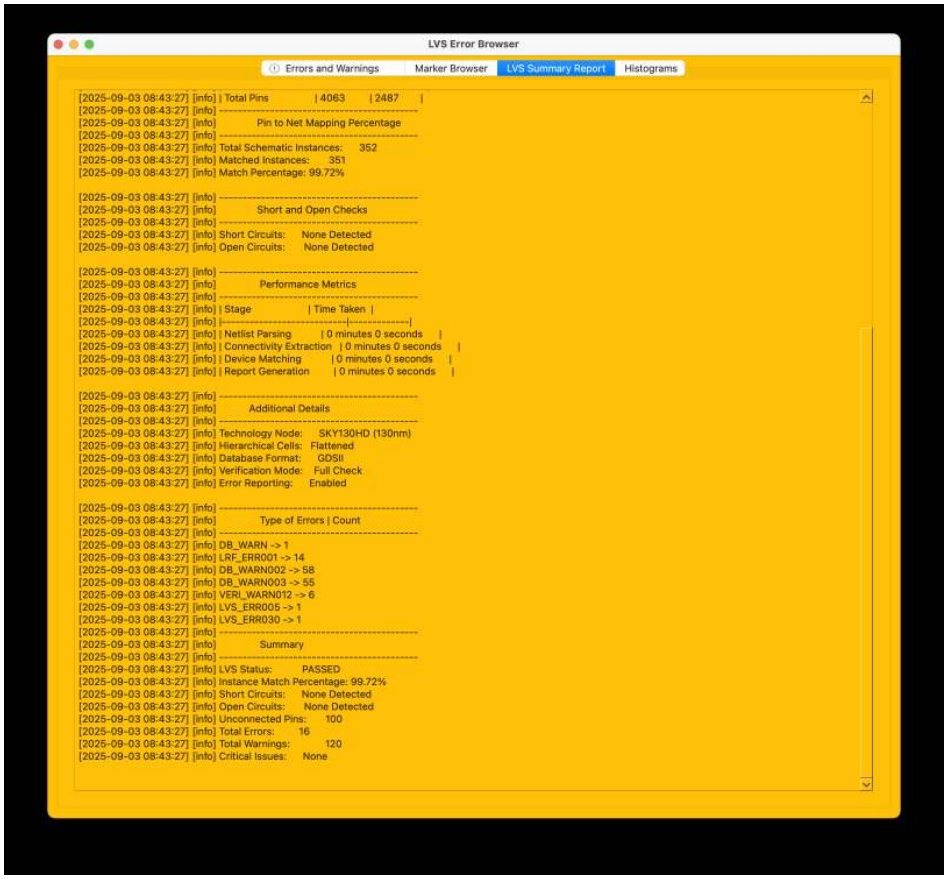

Figure 3. LVS Summary Report: short and open counts, device and instance match percentages, totals and per-stage timing.

AN-05 Waiver Management and Incremental Sign-off

Problem	Applies to
Known, reviewed violations must be suppressed without creating a mechanism that can hide new ones.	DRC, LVS, ERC. Any project reaching sign-off.

The danger with waivers is not that they suppress violations — that is their purpose — but that a waiver written for one revision can silently continue to suppress a different violation in the next. LVR addresses this by binding each waiver to a SHA-256 hash of the geometry it was written for. When that geometry changes, the waiver stops applying automatically and the violation reappears.

Setting up

```
set_lvs_waiver_file ./waivers/project.db
set_lvs_enable_waivers 1
set_lvs_waiver_expiry_check 1
set_lvs_waiver_audit_trail 1
set_lvs_waiver_spatial_filter 1
set_lvs_waiver_owner rpatel
```

Setting	Why it matters at sign-off
`set_lvs_waiver_expiry_check 1`	Reports lapsed waivers, so a waiver granted as temporary cannot quietly become permanent
`set_lvs_waiver_audit_trail 1`	Records which waiver suppressed which violation — the record a sign-off review needs
`set_lvs_waiver_spatial_filter 1`	Confines a waiver to the region it was written for, so it cannot travel to an unrelated part of the die
`set_lvs_waiver_owner`	Attaches accountability to each waiver
`set_lvs_waiver_wildcard 1`	Convenient during development. Avoid at sign-off: a broad pattern can suppress violations nobody reviewed

Waiving specific items

```
lvs_waiver_macro_net sram_1024x32 VDDA
lvs_waiver_netlist_net VDDA_NOCONN
```

Making a run reproducible

```
set_lvs_signature_file ./out/run.sig
set_testmode 1
disable_logger_timestamp
```

The signature file records a hash of every input, so a later run can prove it operated on identical data. Combined with test mode and timestamp suppression, this makes two runs of the same script byte-comparable — which is what turns 'the results changed' into a question with an answer.

Expected behaviour to communicate to reviewers

A waiver that stops suppressing a violation after a layout edit is working correctly, not failing. Reviewers unfamiliar with hash-bound waivers sometimes read the reappearance as a regression. It is the opposite: it is the mechanism catching a change that a hand-maintained waiver list would have hidden.

AN-06 Runtime Tuning for Full-Chip Runs

Problem	Applies to
A full-chip run takes too long, or does not speed up when more cores are made available.	All engines. Designs large enough that runtime is a schedule constraint.

Step 1 — measure before changing anything

```
set_lvs_report_timing 1
enable_logger_timestamp
```

Read the per-stage timing in the Summary Report. Tuning the wrong stage is the most common way to spend effort without changing the result.

Step 2 — check the geometry tier

This is the largest single lever, and the one most often overlooked. See AN-02. If runtime changed by an order of magnitude with no change in marker counts, start here.

Step 3 — balance regions against cores

```
set_num_cores 16
set_num_regions 64
set_halo_width 5000
```

Symptom	Cause	Action
Runtime does not improve with more cores	Too few regions; workers idle	Raise the region count above the core count
One region dominates the total	Uneven density; one block far denser than the rest	Raise the region count further so the dense area subdivides
Missing violations near region boundaries	Halo smaller than the largest interaction distance in the deck	Raise <code>set_halo_width</code>. This is a correctness problem, not a performance one — fix it before tuning anything else.
Memory exhaustion	Too many regions, or caching too aggressive	Lower the region count; set <code>set_lvs_memory_limit_mb</code> so the engine spills rather than being killed

Step 4 — LVS-specific settings

```
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 that matter most on a design dominated by repeated standard cells. Hierarchical mode avoids re-extracting an identical cell once per placement, which on a large digital block is the difference between hours and minutes.

Step 5 — PEX-specific settings

```
set_pex_num_threads 16
set_pex_inter_net_coupling_only 1
set_pex_max_coupling_dist_um 2.0
set_pex_rc_reduction 1
set_pex_skip_power_nets 1
```

Coupling distance dominates PEX runtime: every conductor pair within the distance must be considered. Setting it beyond the range where coupling is physically significant is wasted work. Skipping power nets greatly shrinks a signal-only extraction, but do not use it if IR drop analysis is wanted.

Step 6 — turn diagnostics back off

Verbose logging, per-node IR output, per-segment EM output and raw RC dumps all cost work proportional to design size. Leaving them enabled in a production script is a quiet, persistent runtime tax that nobody attributes correctly because it was introduced during a debug session months earlier.

AN-07 Triaging a Run with Thousands of Markers

Problem	Applies to
A run reports a very large number of violations and there is no obvious place to start.	DRC, LVS, ERC, density. Deck bring-up and first runs on a new block.

The instinct to open the first marker and work down the list is wrong. Large marker counts are usually a small number of systematic causes, and the goal of triage is to find those causes rather than to inspect instances.

Step 1 — read the categories, not the markers

The Errors and Warnings tab groups by error identifier. The shape of the grouping answers the first question immediately: a few categories with enormous counts means a small number of systematic problems; many categories with small counts means genuinely independent issues.

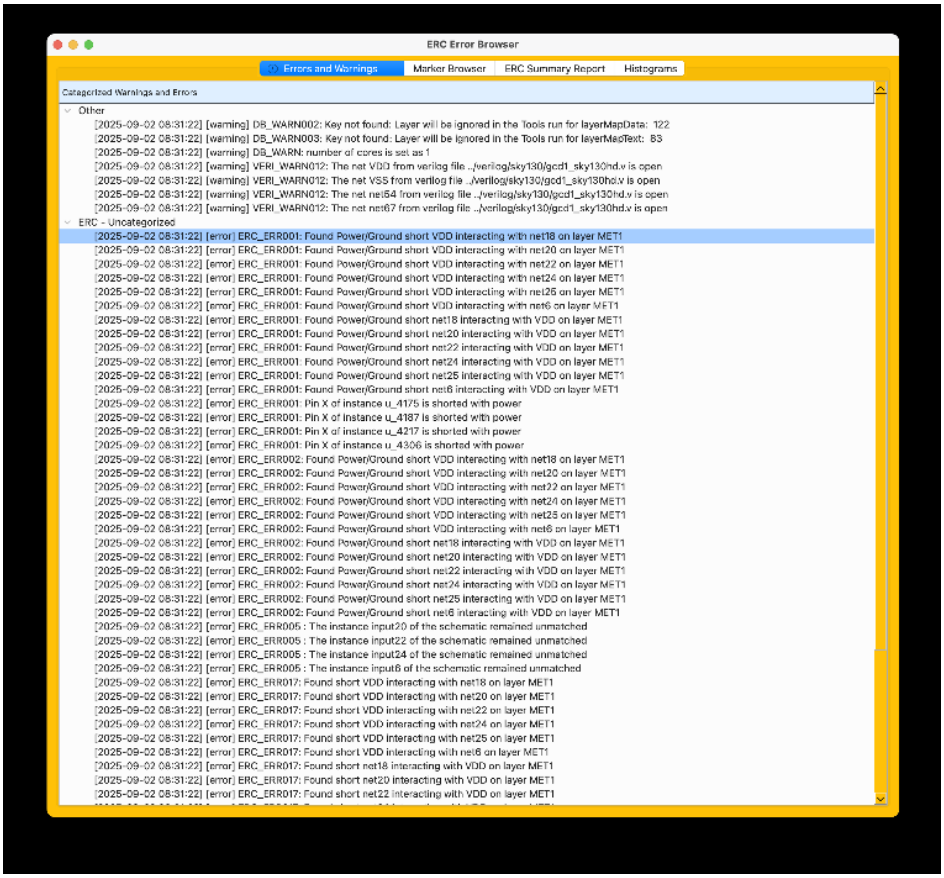

Figure 4. Errors and Warnings tab. Grouping by category is what makes a large marker count reviewable.

Step 2 — read the heat map

The Histograms tab plots marker density spatially. A single hot region usually means one block, or one repeated cell, accounts for most of the count — in which case fixing one cell fixes thousands of markers.

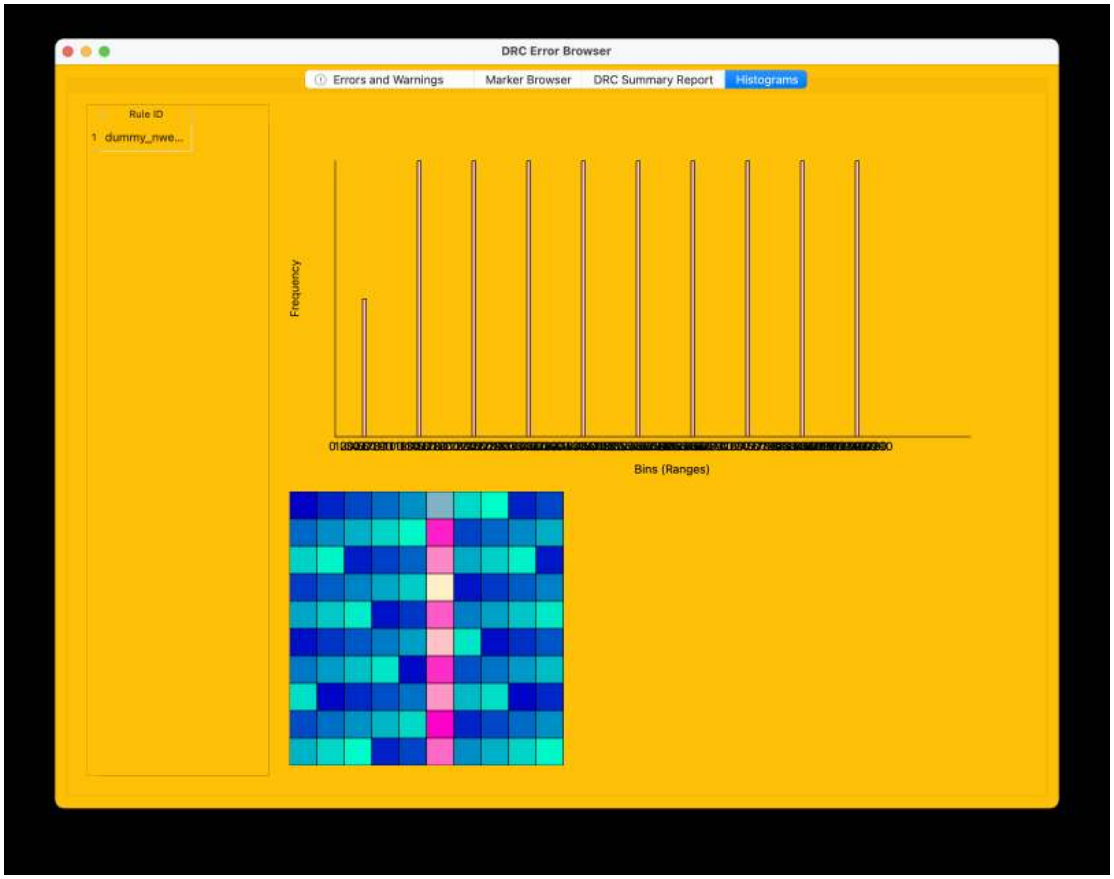

Figure 5. Histograms tab. Frequency by bin above, spatial density map below.

Step 3 — distinguish real from artifact

Pattern	Likely cause	Confirm by
One rule accounts for nearly everything	A rule referencing the wrong layer, or a threshold in the wrong units	Reading the rule and checking the layer it references is populated
Whole rules report exactly zero	A layer is empty	Checking the layer map and confirming the declaration file was read first
Count stops at a round number	A marker cap is active	Checking <code>drc_max_error</code> and its equivalents
Violations only at region boundaries	Halo too small	Raising <code>set_halo_width</code> and re-running
Every protected net flagged for antenna	Diodes not extracted	Setting <code>set_lvs_extract_diode 1</code>
Huge LVS mismatch count	One or more shorts merging nets	Fixing shorts first, then re-running

Step 4 — bound the loop while iterating

```
drc_max_error 100
```

A cap makes the debug cycle fast. Remove it before producing any number that will be compared against a golden database — a capped run reports a subset, and comparing a capped count against an uncapped one is meaningless.

Step 5 — preserve results for later review

```
set_lvs_enable_db_export 1
set_lvs_db_file ./out/run.db
set_lvs_report_coordinates 1
set_lvs_enable_marker_file 1
set_lvs_marker_file ./out/run.markers
```

The results database keeps full marker detail so a review session can be resumed later without re-running. On a run that takes hours, this is the difference between reviewing once and re-running for every question.

AN-08 Cadence Virtuoso Integration

Problem	Applies to
Designers work in Virtuoso and want violations to appear on the canvas they are already editing, rather than in a separate tool.	DRC, LVS, ERC. Analog and mixed-signal flows.

LVR pushes markers into a running Virtuoso session as dbCreateMarker overlays through a SKILL callback. Violations are then navigable in the layout editor the designer already has open.

Configuration

```
set_lvs_skill_integration 1
set_lvs_virtuoso_markers 1

set_lvs_virtuoso_lib myLib
set_lvs_virtuoso_cell DiffAmp
set_lvs_virtuoso_view layout

set_lvs_skill_callback lvrMarkerSelect

set_lvs_marker_file ./out/lvs.markers
set_lvs_report_coordinates 1
```

Checking what is on screen, not what is on disk

```
set_lvs_gds_export_before_run 1
set_lvs_cdl_export_before_run 1
```

These export the current cellview and schematic immediately before the run, so the check operates on the designer's live edits rather than on a file saved some time earlier. Without them, the most confusing possible outcome occurs: a violation that was just fixed continues to be reported.

The iteration loop

#	Step	Where
1	Run the check with export-before-run enabled	LVR, batch or Tcl panel
2	Markers appear as overlays on the layout canvas	Virtuoso
3	Select a marker; the SKILL callback fires	Virtuoso
4	Edit the geometry	Virtuoso, or the LVR AMS Layout Editor
5	Re-run	LVR

Where the fix is small and geometric, the AMS Layout Editor inside LVR closes the loop without leaving the tool at all. See the GUI Reference, Chapter 8.

AN-09 Antenna and Electrical Rule Checks on Analog Mixed-Signal Designs

Problem	Applies to
An AMS block reports antenna and ERC violations that appear implausible.	ANTENNA, ERC. Analog, mixed-signal and I/O blocks.

AMS blocks fail antenna and ERC checks for reasons that rarely apply to digital blocks: protection devices the checker does not recognise, supply domains it does not know are separate, and bulk connections it was never told to extract.

Antenna: recognise the protection

```
init_tool_db ANTENNA
set_technode generic12
set_top ams_top
read_svrf ../rule/foundry_deck/tech_antenna.drc.svrf
read_gds ../gds/ams_top.gds

set_lvs_extract_diode 1
set_lvs_use_svrf_device_rules 1

load_db ANTENNA
antenna ams_top
set_report_file_antenna ./out/antenna.rpt
report_antenna
```

Without diode extraction, protection diodes are invisible to the checker and every protected net reports a violation. This produces exactly the pattern that looks like a serious layout problem and is a one-line setup fix. If nearly every gate in an I/O ring is flagged, check this first.

ERC: declare every supply domain

```
init_tool_db ERC
set_top ams_top

set_lvs_power_pin VDD
set_lvs_ground_pin VSS
set_lvs_power_pin1 VDDA
set_lvs_ground_pin1 VSSA
set_lvs_power_pin2 VDDIO
set_lvs_ground_pin2 VSSIO

set_lvs_extract_bulk_terminal 1
set_lvs_extract_nwell_terminal 1
set_lvs_extract_sub_terminal 1

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_power_short 1
set_erc_check_undriven_power 1
set_erc_check_multi_driver 1
set_erc_report_info 1

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

Check	Why it matters in AMS
`set_erc_check_power_short`	Multi-supply AMS blocks have several domains that must stay isolated. A short between them is among the most damaging errors to reach silicon, and geometry checking alone will not find it.
`set_erc_check_floating_bulk`	Analog devices frequently have bulk connections drawn separately from the source. A missed bulk tie leaves the body at an undefined potential and risks latch-up.
`set_erc_check_multi_driver`	Legitimate in tristate and open-drain structures, which AMS blocks use routinely. Expect findings here and review rather than assume they are errors.
`set_erc_check_undriven_power`	Catches a supply that was routed but never connected to a pad or regulator.

Reading the result

Review ERC in the connectivity view, where shapes are coloured by net rather than by layer. A short between two domains then appears as a single connected region in a colour that should not exist — far easier to see than reading a net list.

Figure 6. Connectivity view. Shapes coloured by net; the Selection Panel reports the nets and pins involved in the highlighted region.

Closing the loop

AMS fixes are usually small and local: a missing bulk tie, a diode to add, a strap to extend. The AMS Layout Editor makes these directly, and the check re-runs on the edited database without a round trip through a separate layout tool. See the GUI Reference, Chapter 8.

AN-10 ECO Verification and Revision Comparison with XOR

Problem	Applies to
A layout revision must be proven to contain only the intended changes.	XOR, FASTXOR. ECO cycles, fill insertion, metal-only revisions, tape-out sign-off.

XOR reports every region present in one layout and absent from the other. It is the only check that answers the question 'what actually changed', which is the question an ECO review needs answered.

Screening then confirming

Use the two engines in sequence rather than choosing between them.

Stage	Engine	Answers	Cost
Screen	`run_fastxor`	Which cells differ at all	Low; run across everything
Confirm	`run_xor / xor_check`	Exactly what differs, with precise geometry	Higher; run only on cells the screen flagged

Screening pass

```
init_tool_db FASTXOR
set_technode sky130
set_top gcd1_sky130hd
set_xor1_gds ../gds/revA.gds
set_xor2_gds ../gds/revB.gds
load_db FASTXOR
run_fastxor
set_report_file_fastxor ./out/fastxor.rpt
report_fastxor
```

Confirming pass

```
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
```

set_gui_xor_result 1 loads difference regions into the graphics view as markers, so they are navigated exactly like DRC violations: select a row, the region centres in the canvas.

Reviewing the differences

An ECO review is a classification exercise. Every difference belongs in one of three buckets, and the review is complete when the third is empty.

Bucket	Meaning	Action
`Intended`	The change the ECO was supposed to make	Confirm it matches the ECO description
`Consequential`	A change that follows necessarily from the intended one — fill adjusted around a moved wire, for example	Confirm the mechanism, then accept
`Unexplained`	Neither of the above	Investigate. Sign-off should not proceed while this bucket is non-empty.

Common sources of unexpected differences

Source	Signature
`Fill regenerated between revisions`	Large numbers of small differences on fill layers only, spread evenly across the die
`Different writing tool or version`	Differences concentrated at cell boundaries, or systematic off-by-one-database-unit edges
`Hierarchy flattened on one side`	Differences following cell outlines
`Layer map mismatch`	A whole layer present on one side and absent on the other
`Different top cell`	Everything differs, or the run reports almost nothing

Recording the result

```
set_testmode 1
disable_logger_timestamp
```

For a tape-out record, run with these enabled so the XOR report is byte-comparable against a later re-run. Combined with a signature file, this establishes that the layout signed off is the layout that was checked — which is the claim an ECO sign-off is actually making.