Layout Verification Reporter (LVR) GUI User Guide

Ramtera Inc

December 21, 2025


LVR
Layout Verification
Reporter

Contents

Copyright

Ramtera Design Automation, Inc. 16192 Coastal Highway, Lewes, Delaware 19958, County of Sussex, USA.

Figure 1: Enter Caption

1 LVR GUI Usage Guide

This section describes the features available with the Graphical user interface of LVR. Here we take the example of LVS tool of LVR, instead of going over all the graphics features with other tools of LVR.

1.1 Input Configuration

1.1.1 Run Data: Run Settings

This section allows users to configure execution parameters as an example for LVS (Layout Versus Schematic) runs in LVR. It ensures that users can control runtime limits, parallelism, and top-level hierarchy before launching LVS. **Macro Timeout (sec)** defines the maximum time allocated to resolve large macro blocks individually. This helps prevent stalling due to complex macro-level mismatches during connectivity resolution. The default macro timeout is set to 1 second, but it can be increased based on design size or complex-

ity. **Instance Matching Timeout (sec)** sets an upper limit for the instance-to-instance netlist comparison stage. A value of 3600 seconds (1 hour) is typically sufficient for large hierarchical designs. **Number of Cores** allows users to select how many CPU cores to allocate to the LVS run. This facilitates multithreaded execution, significantly speeding up processing for large designs. Users should choose a value based on system capability and license limits (e.g., max 16 or 32). **Top Cell Name** specifies the name of the hierarchical root cell for layout and schematic comparison. This field is mandatory and must match the top cell present in both layout and netlist databases. If the top cell name is incorrect, LVS will terminate with an error message. The yellow background indicates editable runtime fields specific to LVS control, distinct from input/output paths. Navigation buttons on the left panel allow users to move between configuration pages. The **Cancel** button discards changes, while **Apply** saves them temporarily without running LVS. **Submit** saves settings and starts the LVS process with the configured parameters. Each setting is validated internally to ensure correctness before launching LVS. This configuration page is especially useful for debugging and tuning LVS performance. Users can update values between runs for iterative optimization based on design size and mismatch complexity.

Figure 2: Enter Caption

1.1.2 Rules Tab

The **Rules** section is where users select technology-specific rule files for LVS verification. The **Technology** dropdown lists available technology nodes (e.g., `sky130`, `asap7`, etc.). Selecting the correct technology ensures proper interpretation of layers and devices. An incorrect technology may lead to false errors or mismatches in LVS. The **LRF File** (Layout Rule File) defines how layout and netlist devices are recognized and mapped. This `.pvs` or `.lrf` file includes layer mapping, device recognition, and net extraction rules. Clicking the **...** button allows users to browse and select the appropriate LRF file from disk. Paths can be absolute or relative to the run directory. This step is critical for proper LVS functioning across multiple technology nodes. Make sure the selected technology matches the input layout and schematic data. Misaligned technology and rule files can result in high error counts or failed runs. The selected rules remain cached until explicitly changed or cleared by the user. Apply changes using the **Apply** button to save without starting LVS. Click **Submit** only after all tabs are configured correctly. The configuration file supports

multiple versions, and LVR validates format compliance. Rule files are parsed before execution to ensure syntactic correctness. Errors in the rule file are reported in the GUI error log section. Rules tab must be configured before submitting the run to avoid tool termination. This section enables multi-node support for advanced verification workflows.

Figure 3: Enter Caption

1.1.3 Input Tab

The **Input** section gathers the files needed for LVS comparison. **Module-SPICE File** refers to the flattened SPICE netlist generated from layout. This file should include extracted parasitics and proper instance hierarchy. **Verilog Netlist** provides the reference RTL or gate-level source for comparison. This file must match the hierarchy used in layout netlist extraction. **Macro CDL Library** is used to resolve external macro references during LVS. It ensures that black-box or hard macros are matched and verified correctly. **GDS File1** refers to the physical layout database in GDSII format. The layout file must contain the top cell and all child instances. Clicking **...** next to each file opens a browser to select the appropriate input. Absolute or relative paths can be used depending on user preference. Ensure consistency between the SPICE and Verilog top-level modules. Missing files or unmatched macros will result in LVS errors or termination. Use **Apply** to validate and save inputs before launching the run. The GDS file and netlists should be pre-verified for syntax and completeness. This section is crucial for ensuring the integrity of LVS input

data. Misconfigured inputs can cause false errors or missed shorts/opens. The yellow background distinguishes this section clearly for usability. All fields must be populated before running LVS to avoid tool exit. Validated input paths are cached between sessions for convenience.

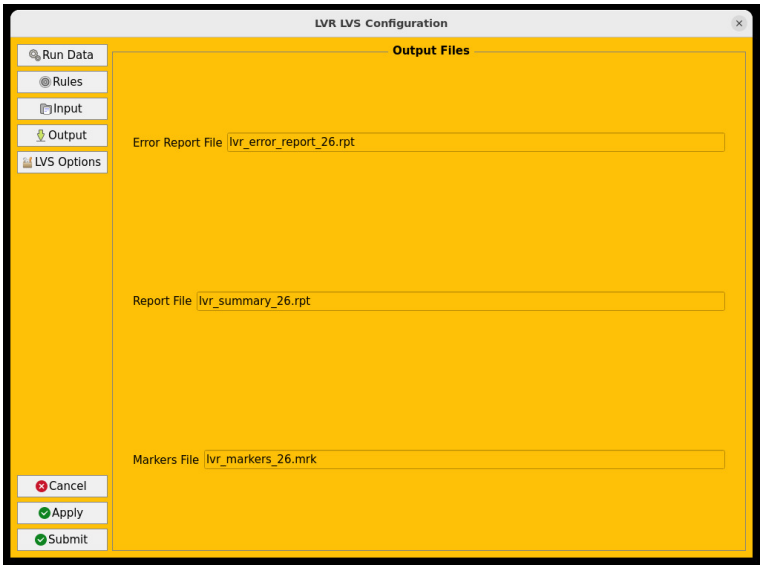


Figure 4: Enter Caption

1.1.4 Output Tab

The **Output** section defines where LVS results and reports are stored. **Error Report File** stores detailed error logs generated during LVS. It includes mismatches, shorts, opens, and macro failures in structured format. **Report File** provides a summary of LVS results including pass/fail status. This file is suitable for quick reviews and automation scripts. **Markers File** logs layout coordinates of error markers, helpful for GUI debugging. Markers can be visualized later using the LVR viewer or external tools like KLayout. Users can specify filenames or use defaults that include a timestamp or ID suffix. Click **...** to browse or manually enter the desired output file name. Ensure write permissions exist in the output directory. Overwriting existing files without notice is prevented via internal checks. Reports are generated only after successful completion of LVS. Incomplete runs will leave some files empty or missing. Keep outputs in organized directories per design or run ID for traceability. Output filenames can also be used for regression tracking. Markers are essential for visually resolving layout-level issues. Use **Apply** to store output file selec-

tions without launching LVS. Reports can be browsed in the LVR GUI post-run using the error browser. This tab simplifies debugging and helps in run result documentation. Ensure consistent naming if scripting post-run analysis workflows.

Figure 5: Enter Caption

1.1.5 Options Tab

The **LVS Options** section allows users to enable or disable specific checks. **Enable Ports** ensures that external ports are considered during LVS. This is critical for top-level connectivity and interface matching. **Use DATC Flow** activates flow compliance with the DATC reference benchmarks. Enabling it aligns output format and checks with public LVS standards. **Check Shorts** validates if unintended connections exist between nets. **Check Open** finds cases where nets are disconnected or incomplete. **Check Macro** inspects macro-level block connections for consistency. **Check Instance Matching** ensures each layout instance matches its schematic counterpart. **Check Unconnected Pins** detects floating or unconnected pins that can cause logic errors. All options are enabled by default for comprehensive LVS coverage. Users may selectively disable options for debug runs or performance tuning. Disabling essential checks may lead to false LVS pass results. These options directly influence runtime and reporting verbosity. Use **Apply** to save changes and **Submit** to execute the run. Options are stored in the configuration and

reused across sessions. This panel allows fine-grained control over the LVS flow. It enables targeted debugging and iteration for mismatched designs. The yellow layout and checkbox format simplify usability and reduce errors. These settings can also be overridden via command-line flags in automated runs. Proper configuration here is crucial for reliable LVS outcomes.

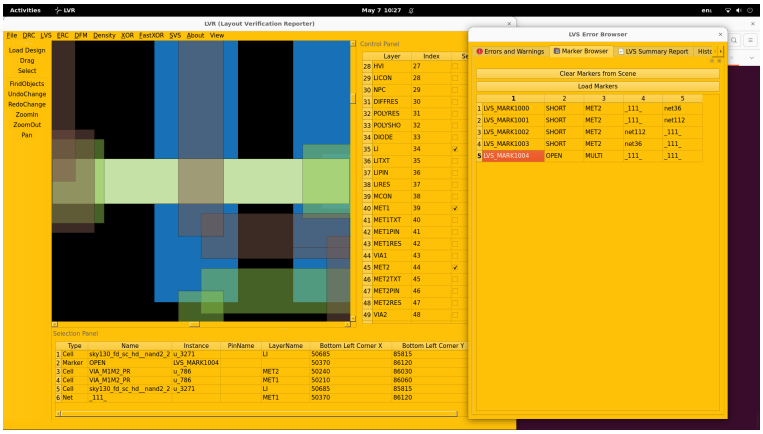


Figure 6: Enter Caption

1.2 Error Browser

1.2.1 Overview Panel and Marker Integration

The main LVS Error Browser panel provides a synchronized view of the layout, control panel, and error markers. The layout window (left) displays the physical design, while the right-hand error browser shows marker data. Each LVS error is listed with a unique marker ID, type (e.g., SHORT, OPEN), layer, and interacting nets. Selecting a marker highlights the corresponding area in the layout and updates the selection panel. The selection panel at the bottom provides detailed instance, pin, and layer information for the selected marker. The layer visibility control (center) allows toggling display of individual metal and via layers. This integration helps users visually inspect shorts, opens, or mismatches with accurate coordinates. Each net in the layout is interactively highlighted, aiding quick debugging of LVS violations. Users can filter layers for clarity, and markers are color-coded by error type. This consolidated view supports rapid identification and resolution of physical LVS issues.

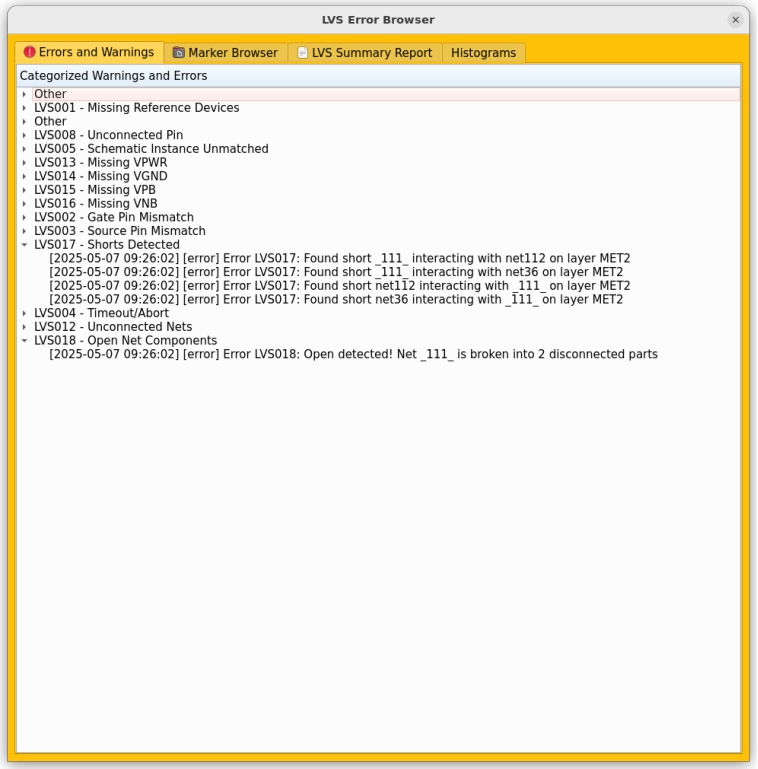


Figure 7: Enter Caption

1.2.2 Errors and Warnings Panel

The **Errors and Warnings** tab categorizes LVS violations using standard error codes. Each error category (e.g., LVS017 - Shorts, LVS018 - Opens) is collapsible and displays sub-errors with timestamps. These logs are automatically grouped for easier navigation during large design analysis. Each message contains detailed context: involved nets, layers, and the nature of the error. Warnings and errors can be reviewed post-run, allowing iterative LVS debug flows. Errors are prefixed with timestamps and log levels (info, error) for traceability. This textual view complements the graphical marker interface. Error types such as ”Missing Reference Devices”, ”Unconnected Pins”, or ”Source Pin Mismatch” are easily browsable. The panel helps engineers

correlate error markers with detailed error descriptions.

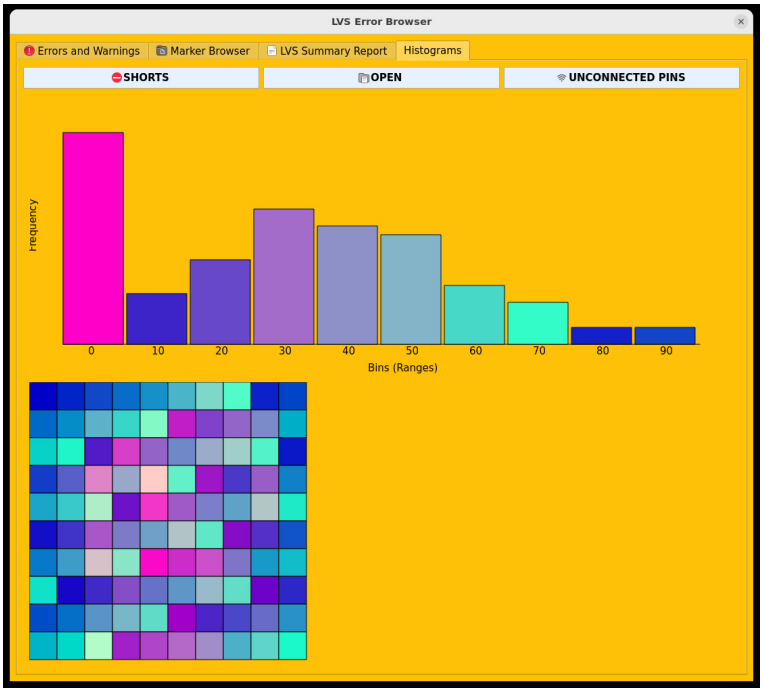


Figure 8: Enter Caption

1.2.3 Histograms Tab

The **Histograms** tab provides spatial distribution of errors across the design. Users can toggle between SHORTS, OPEN, and UNCONNECTED PINS error types. The heatmap at the bottom left shows spatial frequency of errors using color-coded bins. The main bar chart visualizes the count of errors per bin, helping identify error-dense areas. This view enables prioritization of debug regions in large layouts. Each bin corresponds to a section of the layout, and the zoom factor can be configured in Preferences. Hotspots (e.g., high frequency of shorts) are colored in brighter tones like magenta or pink. This statistical view assists in assessing layout quality and patterning issues. Histogram visualization is crucial for batch validation and regression debugging workflows.

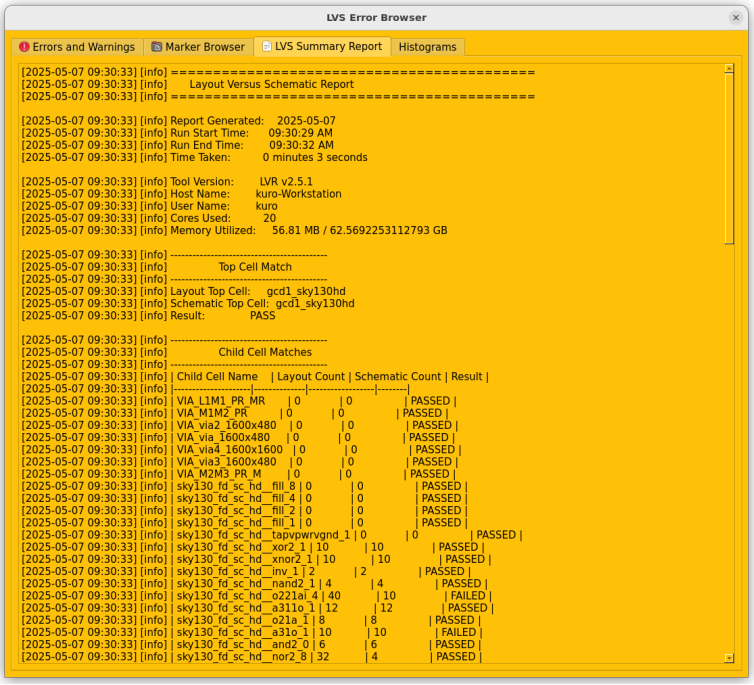


Figure 9: Enter Caption

1.2.4 Summary Report Tab

The **LVS Summary Report** tab displays the full summary log of the LVS run. It contains metadata such as tool version, host, user name, core usage, and memory footprint. The top cell match section shows layout and schematic cell names along with pass/fail status. Child cell matches are listed in a table format with counts and match results. Each cell is checked for equivalence in hierarchy and connectivity. Failed cells are highlighted and must be resolved before achieving a PASS. The summary gives a complete picture of LVS correctness at hierarchical levels. This log is often archived as part of design verification documentation. It can also be used in automation pipelines for pass/fail gating.

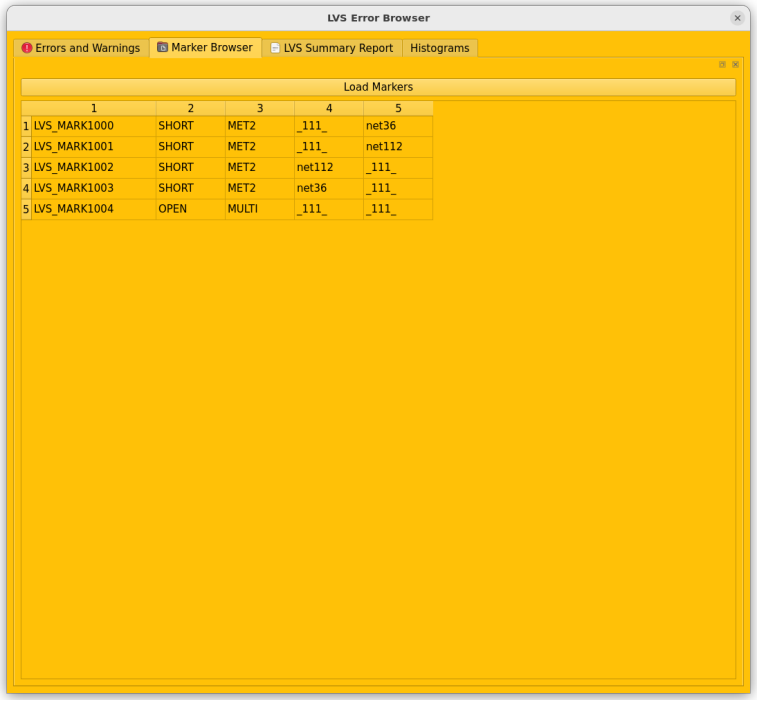


Figure 10: Enter Caption

1.2.5 Marker Browser Tab

The **Marker Browser** tab presents a tabular view of LVS error markers. Each row corresponds to a unique marker, listing marker ID, type (SHORT or OPEN), layer, and nets involved. Users can click on any marker row to highlight its location in the layout viewer. This browser supports sorting and filtering for quick navigation. Markers can be loaded or cleared using the provided buttons. It is designed for rapid pinpointing of problematic layout regions. This tab is ideal for use during interactive layout inspection sessions. It also supports batch mode operation where all markers are loaded from an external file.

1.3 Graphics View

The **Graphics View** in LVR (Layout Verification Reporter) is the central area for layout visualization and graphical interaction. It enables users to examine the physical representation of the design with intuitive panning, zooming, and object selection. As seen in Figure 11, this view displays the layout across multiple layers and supports interactive debugging by highlighting selected cells, nets, and geometries.

The layout canvas is flanked by essential tools such as the Layer Control Panel, Navigation Panel, FindObjects Dialog, and the Selection Panel. These components collectively enable a seamless GUI-based experience for exploring verification errors, inspecting design connectivity, and reviewing geometry across process layers.

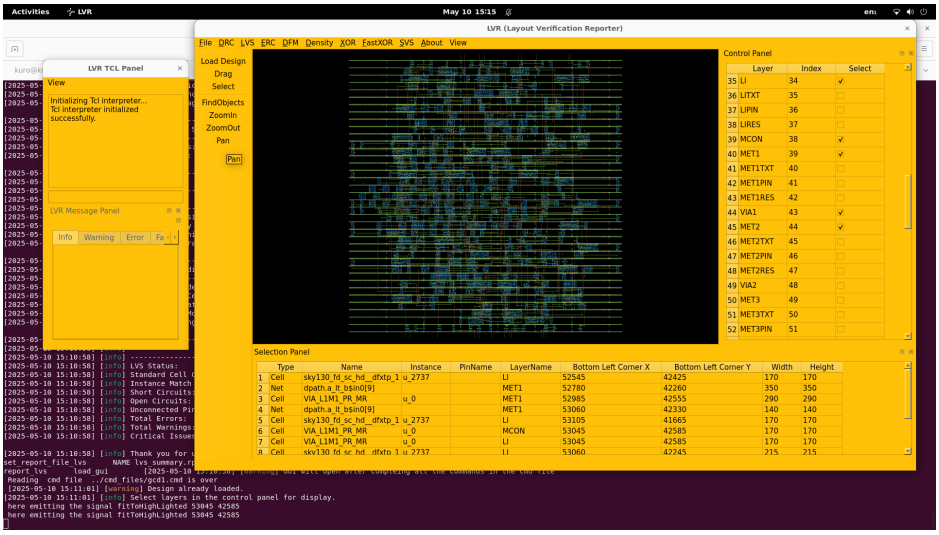


Figure 11: Main Graphics View with active layout display and pan tool

1.3.1 Layer Control Panel

The **Layer Control Panel** (Figure 12) allows users to toggle visibility of specific layers such as LI, MET1, MET2, VIA1, and MCON. Each row in the panel

consists of:

- **Layer**: The name of the layout layer (e.g., MET1, VIA1).
- **Index**: The internal identifier used by the tool.
- **Select**: A checkbox to show or hide the layer in the view.

This selective visibility aids in debugging by isolating problematic layers, helping users focus on shorts, opens, or misalignments at a specific metal or via level.

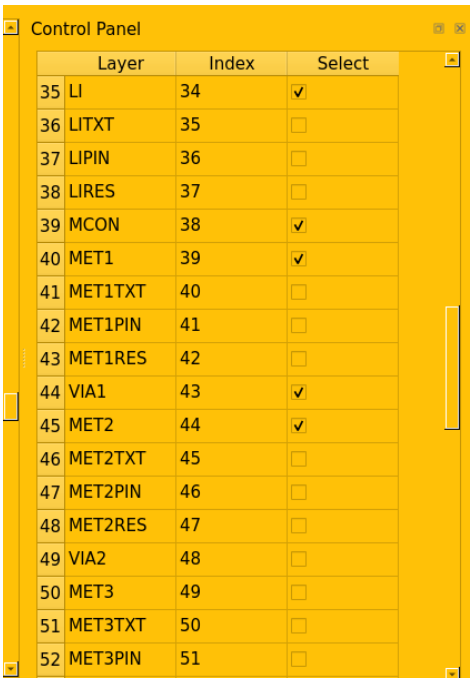


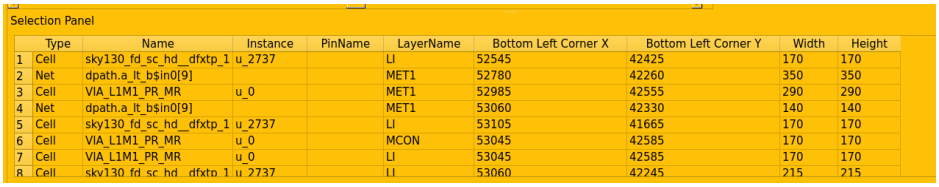
Figure 12: Layer Control Panel

1.3.2 Selection Panel

The **Selection Panel** (Figure 13) shows properties of selected graphical items. Each entry includes:

- **Type**: Identifies if the object is a Cell or Net.
- **Name**, **Instance**, **PinName**: Metadata for the selected element.
- **LayerName**: Corresponding layer.
- **Bounding Box**: Bottom-left coordinates (X, Y), Width, and Height.

This panel helps in verifying if a net or cell instance is correctly placed, sized, and connected.


Selection Panel

	Type	Name	Instance	PinName	LayerName	Bottom Left Corner X	Bottom Left Corner Y	Width	Height
1	Cell	sky130_fd_sc_hd__dfxtp_1	u_2737		LI	52545	42425	170	170
2	Net	dpath.a_lt_b$in0[9]			MET1	52780	42260	350	350
3	Cell	VIA_L1M1_PR_MR	u_0		MET1	52985	42555	290	290
4	Net	dpath.a_lt_b$in0[9]			MET1	53060	42330	140	140
5	Cell	sky130_fd_sc_hd__dfxtp_1	u_2737		LI	53105	41665	170	170
6	Cell	VIA_L1M1_PR_MR	u_0		MCON	53045	42585	170	170
7	Cell	VIA_L1M1_PR_MR	u_0		LI	53045	42585	170	170
8	Cell	sky130_fd_sc_hd__dfxtp_1	u_2737		LI	53060	42245	215	215

Figure 13: Selection Panel displaying highlighted instance `u_0`

1.3.3 FindObjects Dialog

The **FindObjects Dialog** (Figure 14) allows searching for specific nets or instances in the layout. This utility provides two buttons:

- **Find Net**: Zooms into and highlights a given net.
- **Find Inst**: Locates an instance such as a cell and centers it in the graphics view.

It is a powerful feature for quickly navigating large designs, especially when reviewing reported issues.

1.3.4 Search Result View

Figure 15 shows the layout centered and zoomed into instance `u_0` after performing a search via the Find Inst button. This interactive navigation helps in pinpointing the physical location of a logic element reported in error.

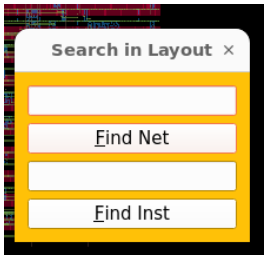


Figure 14: FindObjects Dialog for net/instance search

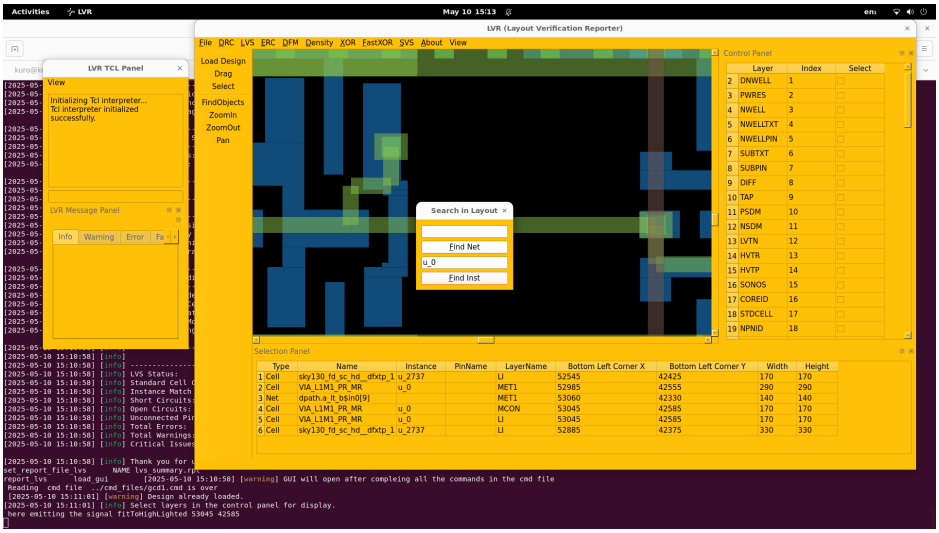


Figure 15: Graphics view zoomed into instance `u_0` from FindObjects

1.3.5 Navigation Panel

Located on the left (Figure 16), the **Navigation Panel** provides push buttons for essential GUI interactions:

- **Load Design**: Load GDS or DEF layout files into the Graphics View.
- **Drag**, **Select**: Enable view manipulation and object selection.
- **ZoomIn**, **ZoomOut**: Adjust zoom level for finer or broader view.
- **Pan**: Move the visible region of the layout while maintaining the zoom

level.

Figure 16: Navigation Panel with layout view tools