Physical Verification System

cadence®

PVS Introduction

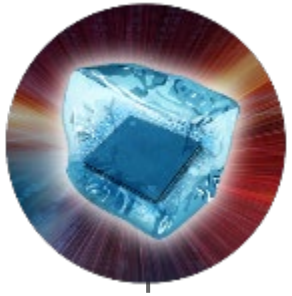
PVS in the Cadence SSV Solution

Tempus
Timing Solution

Quantus QRC
Extraction Solution

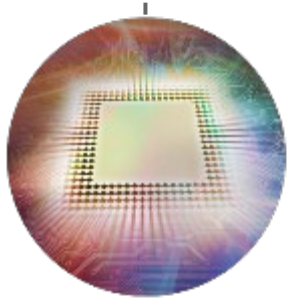
Innovus
Implementation Solution

Joules
RTL Power Analysis Solution



PVS
Verification Solution

Voltus
Power Integrity Solution



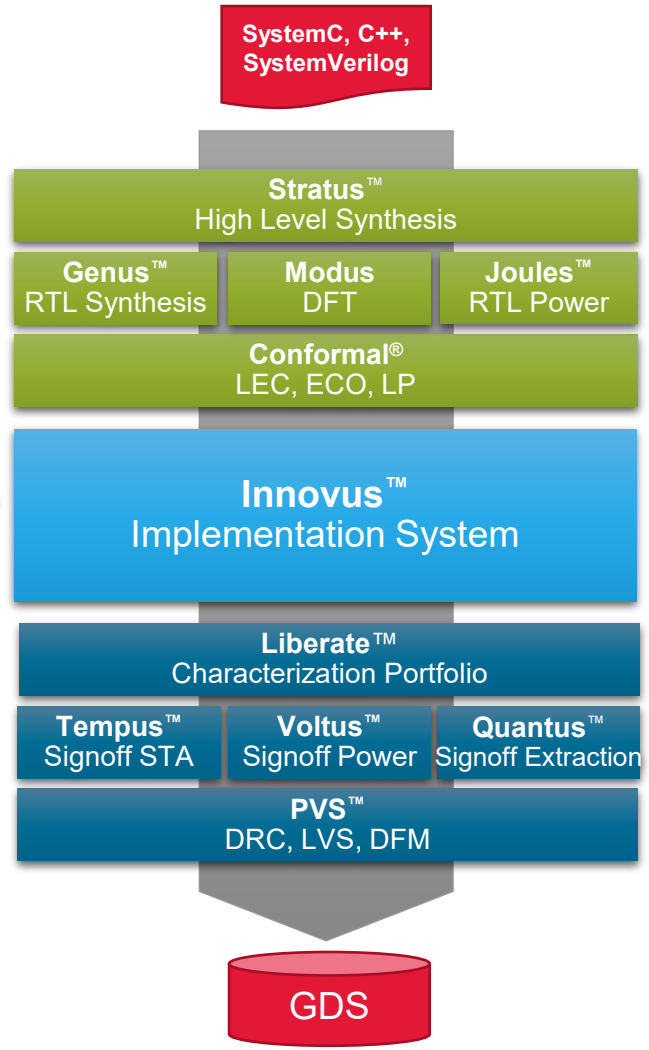
Stratus
High-Level Synthesis Solution

Genus
Synthesis Solution

Modus
DFT

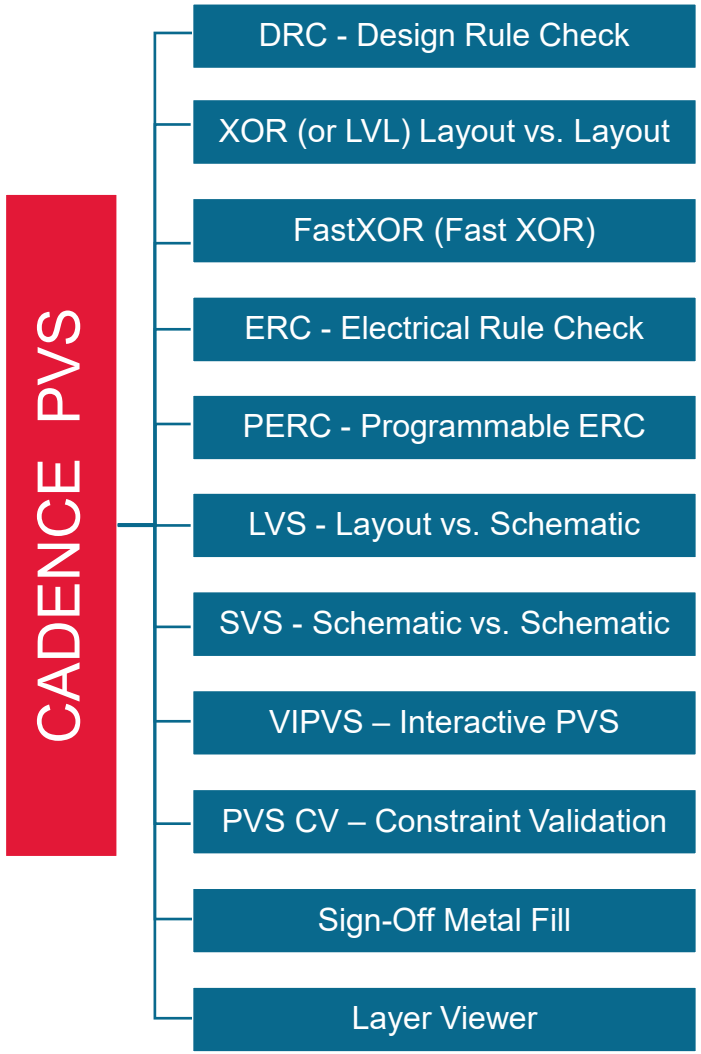
Liberate Trio
Characterization Portfolio

Cadence Full-Flow Digital Solution



Verification in PVS

The chart shows the available verifications in the Cadence Physical Verification System (PVS).



Physical Verification System
Features

- DRC/Fill/LVS Signoff verification
- Seamless integration with Cadence Virtuoso and Innovus™ platforms
- Improved layout productivity
  - Distributed multithreading processing capability – Reduced TaT
  - VIPVS – Correct-by-construction, real-time, in-design DRC solution flow
  - Low transition cost using existing foundry-certified rule decks – Available for all major foundries
- Wide range of process nodes.
  - From planar node to FinFET advanced node design signoff
  - DPT/MPT, 3D-IC and advanced device extraction
- Extends to design reliability checking and Constraint Validation (CV)
  - CV solution for Virtuoso and mixed-signal flow
  - Design for Reliability solution with PVS PERC

How to Set Up PVS?

Software

This is a quick reference on how to set up PVS in your system with relevant PVS paths and variables.

1. Download and install PVS software in your system.
2. Set the relevant PVS paths.

```
set path=(<installation_dir>/bin <installation_dir>/tools/bin $path)
```

3. Install OpenAccess (OA) shared libraries and binaries – several PVS internal tools use OA libraries.
4. For proper distributed processing execution, have the following as your first line in your .cshrc file:

```
if (! $?prompt) exit
```

5. Launch Virtuoso (e.g., IC618), Innovus, or PVS QuickView™ for viewing errors with the error browsers.
6. To preset PVS GUI with the specific default settings, use:
   - Technology setup or PVS pre-trigger function pvsPreFormTrigger.
7. There are two ways of running and using PVS:
   1. Batch mode, using the command-line command – PVS
   2. Interactive mode, using a graphical user interface (GUI) – ***run submit*** GUI

Example .cshrc entries

```
if ( ! $?prompt ) exit
...
...
setenv pvs_install_path /home/user1/pvs
set path = ( $pvs_install_path/bin $path )
...
```

How to Set Up PVS?

Technology Data

Technology data (rule decks etc. for the process) are collected at the technology directory.

1. Define the technology mapping file (typically pvtech.lib), which is a collection of paths to different technology directories.
   - Sample pvtech.lib with techRuleSets file inside the Tech directories tech065 and tech090.

```
DEFINE tech065   /home/alpha/…/…/tech065
DEFINE tech090   /home/alpha/…/…/tech090
```

Technology	ruleSets
Tech090 *or* tech065	default
	OffgridCheck
	antenna

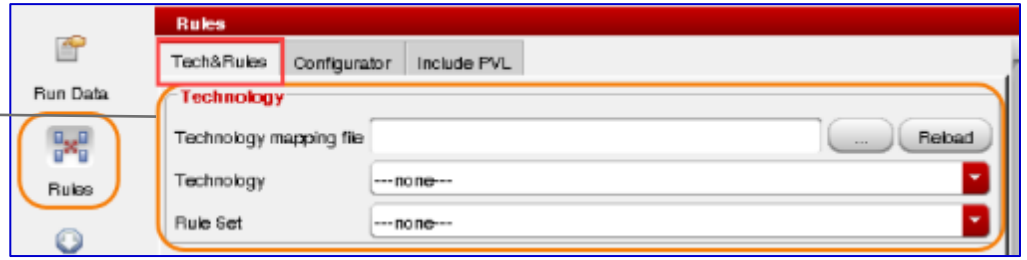
```
++++++ Sample techRuleSets File ++++++++
pvsRuleSet( "default"
(DrcRules "drcMain.rul")
(LvsRules "lvs.rul")                          )
pvsRuleSet( "OffgridCheck"
(DrcRules "offgridcheck.rul" )                )
pvsRuleSet( "antenna"
(DrcRules "offgridcheck.rul" "antenna.rul") )
++++++++++++++++++++++++++++++
```

How to Set Up PVS?

Technology Data (Continue)

2. From GUI: Inside each technology directory, ruleSets define sub-groups of technology data files.
   - PVS ruleSets are defined by techRuleSets files.

From GUI: For any PVS run, the user needs to define a combination of tech mapping file, tech directory & ruleSet.



From the command line, use the technology command:

Format: `technology technologyName [-ruleSet rule_set_name][-techLib path]`

Example: `technology cmos65 -ruleSet "antenna" -techLib "/home/PVS/pvtech.lib"`

Design Rule Checking (DRC)

PVS Design Rule Check

- A PVS job can be started from one of the Cadence backend viewers like Virtuoso, Innovus™ or PVS QuickView – in GUI mode or from a UNIX shell in batch mode.
- To run hierarchical DRC, the minimum data required is listed below:
  - DRC Rule Deck → Foundry or Custom
  - Layout DB → DFII, GDSII or OASIS
- PVS DRC debugging in Virtuoso flow
  - Recommendation: Start with the child cells and progress up the hierarchy.
- PVS Rule decks
  - E.g., DRC rule deck, Antenna rule deck etc.
  - PVS rule decks are coded in PVL (PVL is the native language of PVS).
  - PVS rule decks are available for all major foundries.
  - Make sure that all rule decks are available in the right versions.

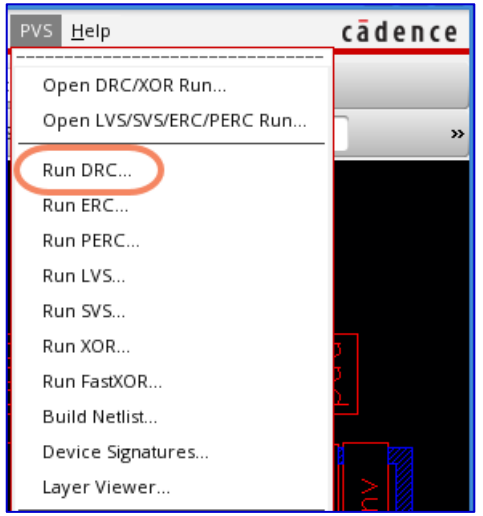
How to Run PVS?

3 Modes

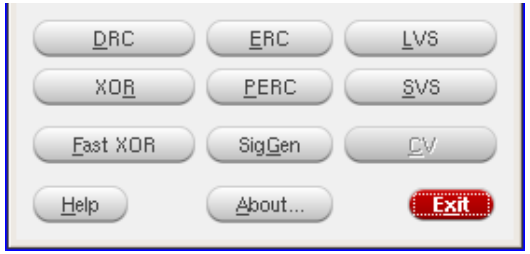
There are 3 modes to run PVS – (1) in Command line (batch) mode, (2) from the GUI integrated to Cadence platforms, or (3) from the Standalone PVSGUI.

1. Command Line – Batch Mode
   - Using **pvs** command from a terminal.
   - Possible to optimize running verification on a large number of cells.
2. Integrated to the Virtuoso/Innovus platforms
   - Interactive graphical interface.
   - Designers can easily verify blocks while completing layouts.
3. Standalone PVS GUI – PVSGUI
   - Invoke standalone PVSGUI with the **pvsgui** command.
   - If **pvsgui** is specified without any option, then it opens the PVSGUI main menu.
   - Clicking menu options to invoke corresponding run forms.

```
pvs -drc [options] ruleFile
```




```
pvsgui &
```



PVS DRC from the Command-Line Options

Batch Mode

```
pvs -drc [options] ruleFile
```

Typical command-line **options** (*applies to DRC/LVL/XOR/FILL flows*):

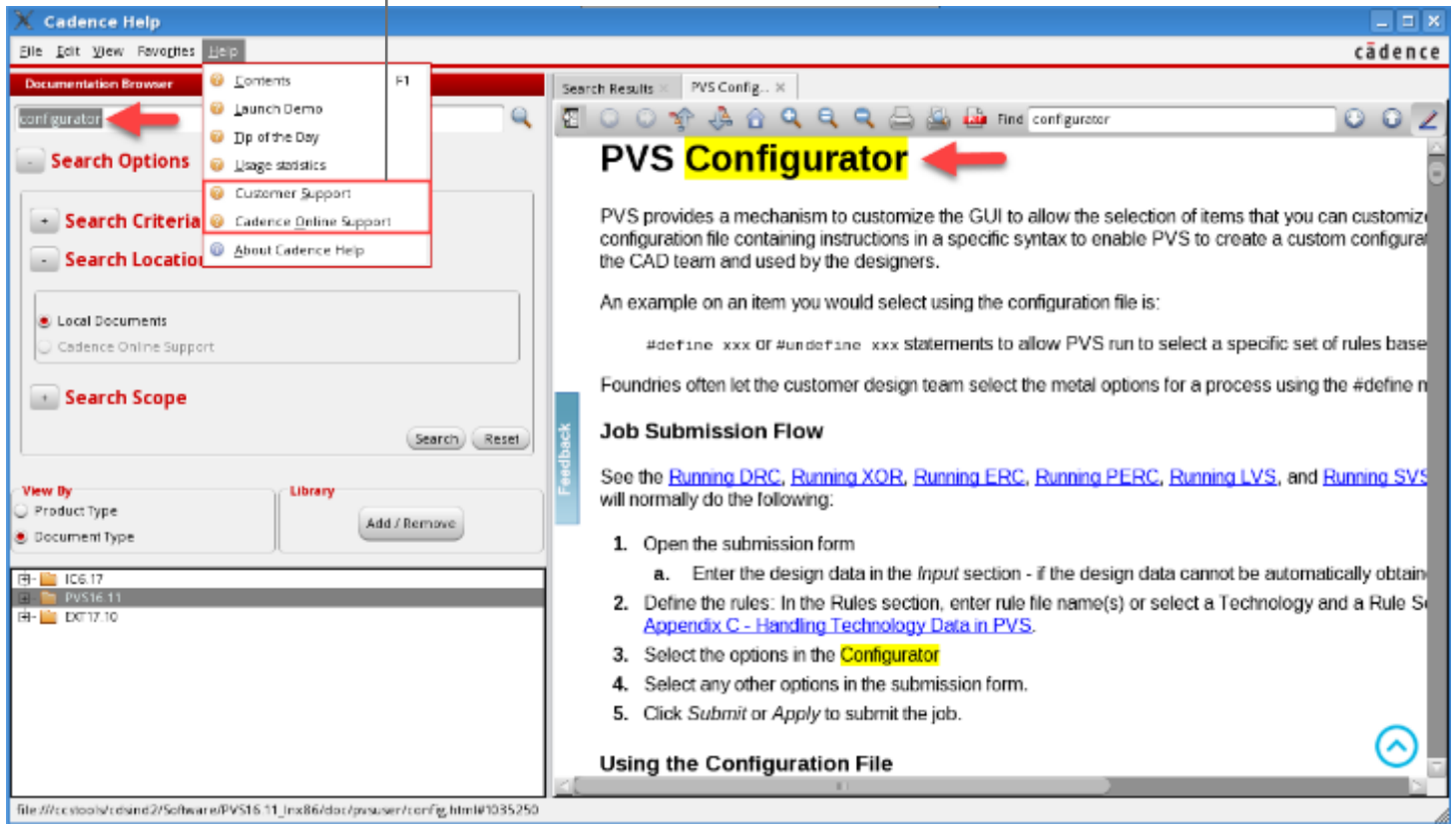
Option	Definition
-gds	Precedes the input GDS filename (use gds or oa)
-oapath	Sets location of the *cds.lib* file (assumes pwd if not specified)
-oalib	Library name (OpenAccess database)
-top_cell	Top cell name
-oaview	View name (OpenAccess database)
-ascrdb | -gdsrdb	Which output format to create – ASCII or GDSII
-h | -help	Displays the available command-line options
-ui_data | -no_ui_data	To specify to generate (or not to) the output data for the error browser
-dp n	Run in local mode, n number of processors

Example: `pvs -drc -oalib tdrc3 -top_cell ecad -oaview layout -ascrdb outFile.asc ../RULES/drc.rul`

How to Search with Keywords Within the Cadence Help GUI?

The Cadence Help utility offers info on topics from within the Virtuoso® environment. It offers options to customize your search criteria, multiple tab support and direct links to Customer Support and COS.

Cadence Help GUI → Help → Customer Support / Customer Online Support



1. Enter cdnshelp in the UNIX prompt and enter – to invoke Cadence Help GUI.
2. Set the scope of search by adding/deleting product libraries.
3. Type the keyword (e.g., Configurator) in search field and enter (or click the Search button). Use multiple search options.
4. Search results appear on RHS window. Finer search options available in RHS – where multiple tabs can open.
5. Option to invoke *COS2.0* or contact *Customer Support*.

Invoking the DRC Run Submission Form

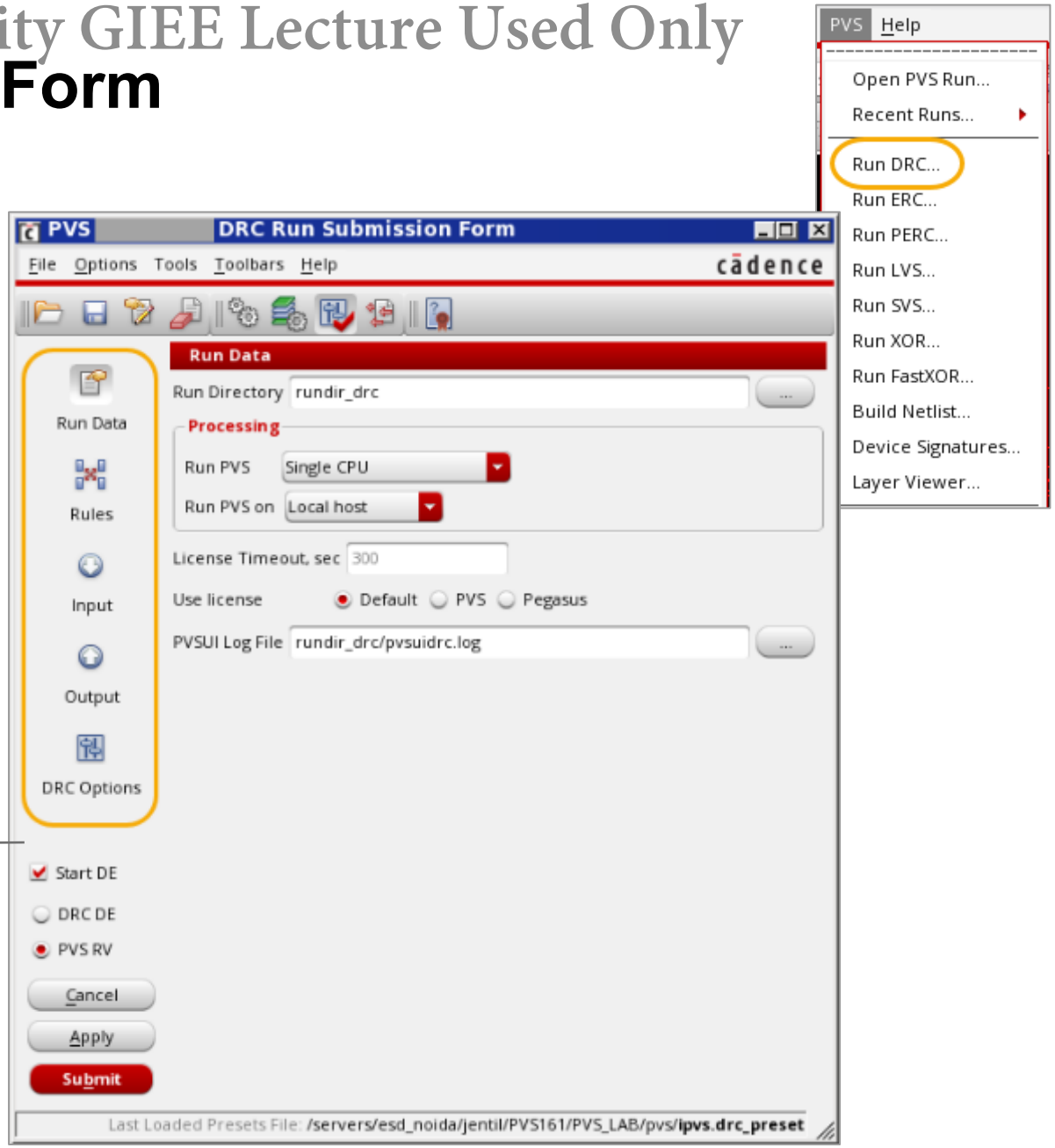
Virtuoso/Innovus/ PVS Quickview → PVS → Run DRC → PVS DRC Run Submission Form.

Standalone PVS GUI → DRC

- This is the starting point for setting up and executing the PVS DRC run.
- This is the basic look of the PVS form for all the verification runs – whether it is DRC/LVS/ERC etc.
- You need to set up this form to submit the DRC run.

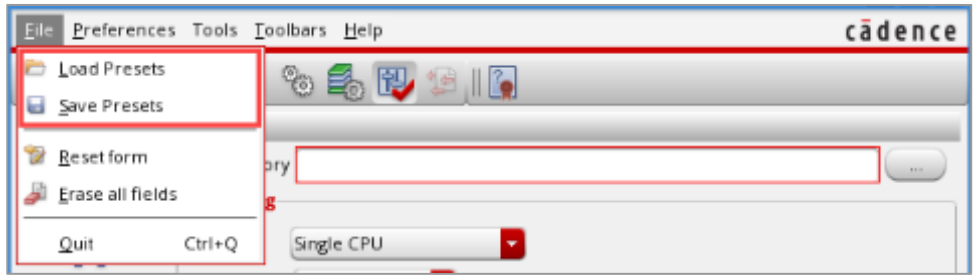
The submission form has five sections:

1. Run Data
2. Rules
3. Input
4. Output
5. DRC Options

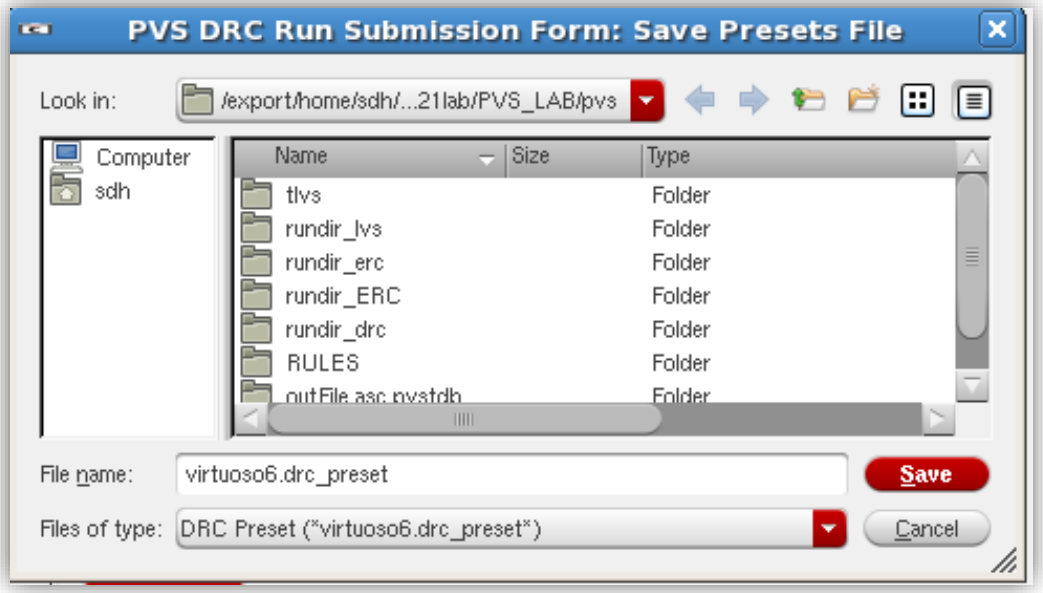


Presets for the PVS DRC Form

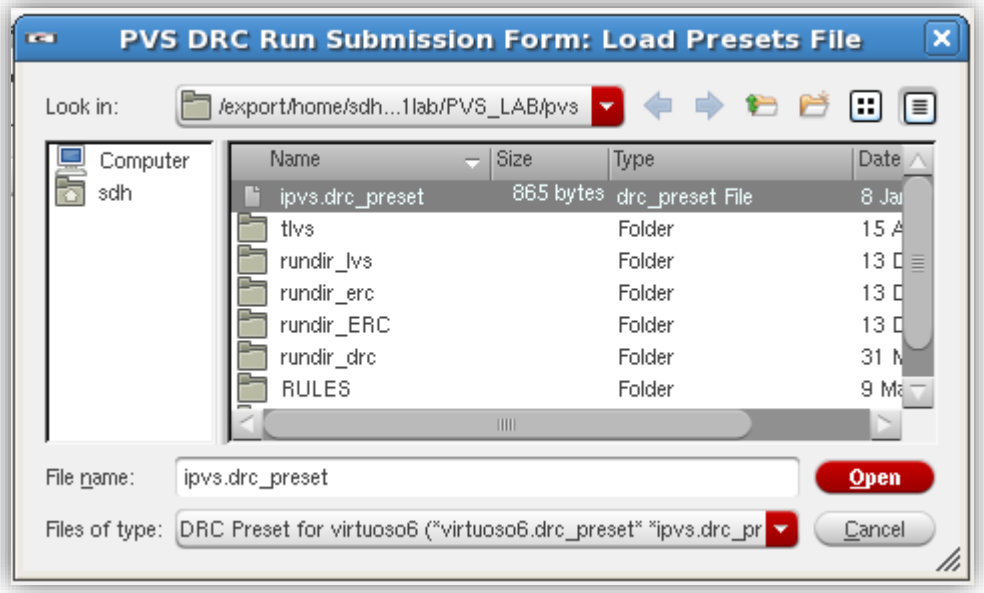
PVS Run Submission Form → File → Save Presets / Load Presets



Save Presets: Saves the values you entered for the DRC run forms to a Preset file

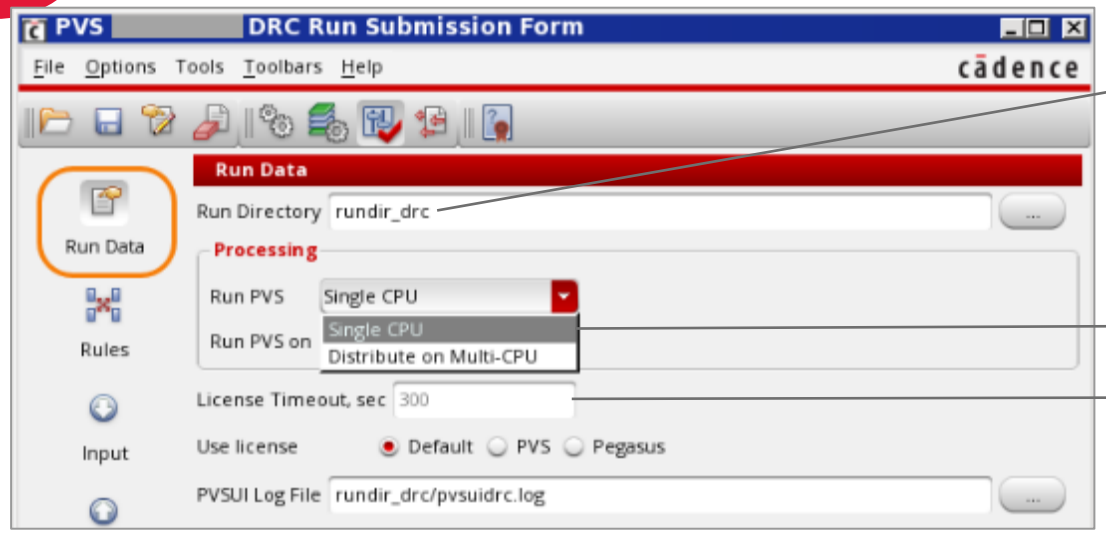


Load Presets: Loads the Preset file



Setting Up Single CPU Processing in PVS GUI

PVS DRC Run Submission Form → Run Data → Run PVS → Single CPU

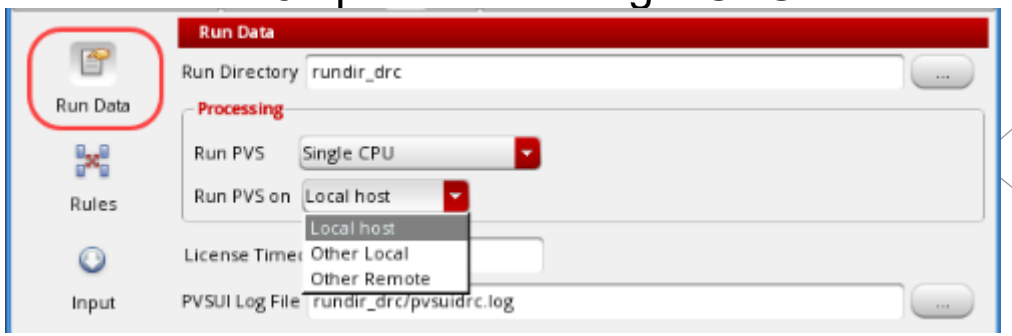


Specify Run Directory

Processing: To *Run PVS* with Single CPU *on* a Local host is default selection
a. Single CPU
b. Distribute on Multi-CPU (Distributed Processing – DP mode)

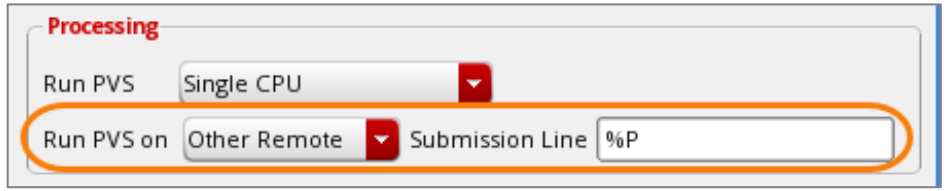
License queuing: If all licenses are in use, then PVS waits for them to free-up. Set timeout period in seconds. Default – 300s.

Run PVS on: 3 options with Single CPU selection:



1. Local host: Run on the local machine (default)

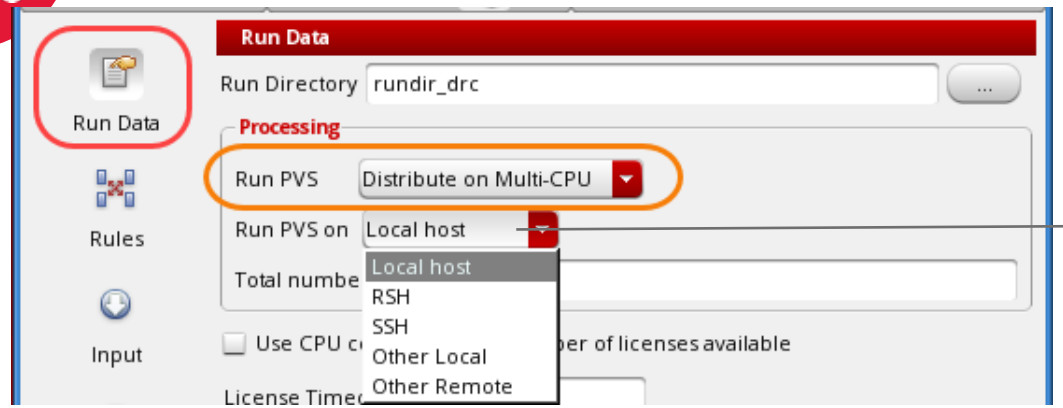
2. Other Local: Run as a part of a batch script on local machine



3. Other Remote: Run from a Remote machine

Setting Up Distributed Processing (DP) in PVS GUI

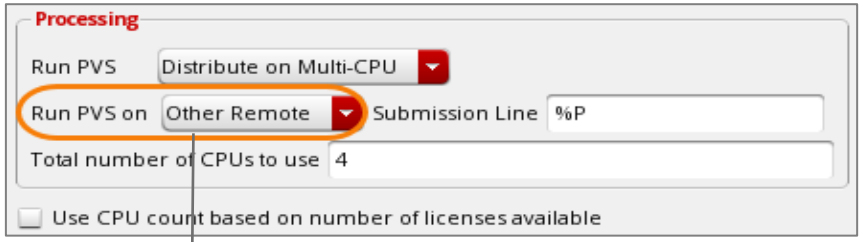
PVS DRC Run Submission Form → Run Data → Run PVS → Distribute on Multi-CPU



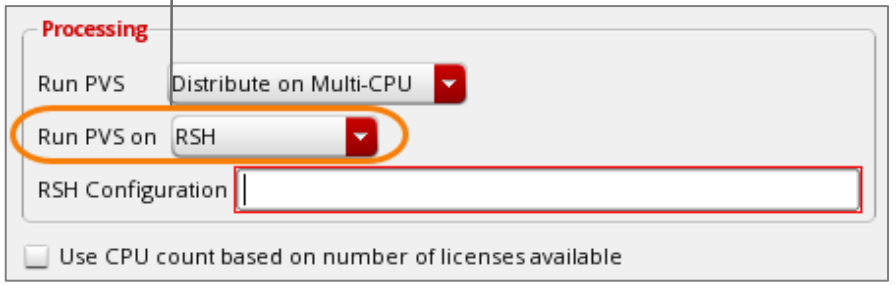
DP mode: Set Run PVS to Distribute on Multi-CPU

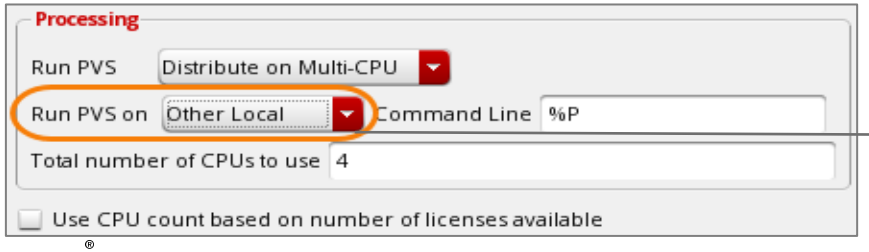
- 5 options with Distribute on Multi-CPU selection
- Enter no of processors at *total number of CPUs to use* textbox
- It should be an integer value between 2 to 128
- Local host: Distributes run to multiple CPUs on the local machine

- RSH / SSH (Only in DP mode): Distributes run in *rsh* (*remote shell*) or *ssh* (*secure shell*) mode



- Other Remote: Submits the run to distributed computing system





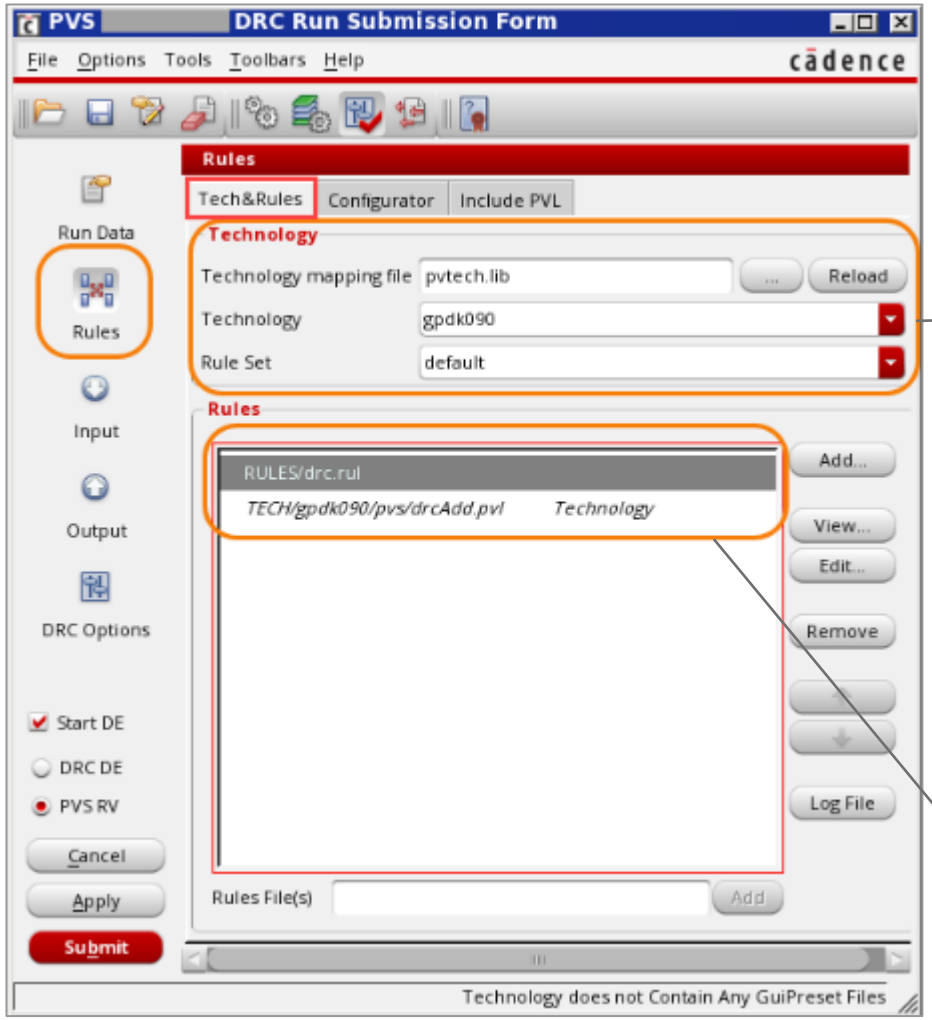
- Other Local: Distributes run as a part of a batch script on local machine
- Multi-host mode defines number of workers and assigns CPUs

DRC Run Submission Form

Rules – Tech&Rules

Select the appropriate technology and rule decks for the PVS run.

PVS DRC Run Submission Form → Rules → Technology & Rules



Sample Tech Map File E.g., `pvtech.lib`	`DEFINE tech065 /home/alpha/…/…/tech065` `DEFINE tech090 /home/alpha/…/…/tech090`

Technology

- Select the Technology mapping file if not pointed by default.
- The Technology field will list directories from the mapping file. Select one.
- Select a Rule Set from the selected Technology directory (ref. `techRuleSets` file).

Sample `techRuleSets` file	`pvsRuleSet( "default"`
	`(DrcRules "drcAdd.pvl")`
	`(LvsRules "lvs.pvl")`
	`)`
	`pvsRuleSet( "antenna"`
	`(DrcRules "offgridcheck.pvl" "antenna.pvl")`
	`)`

Rules

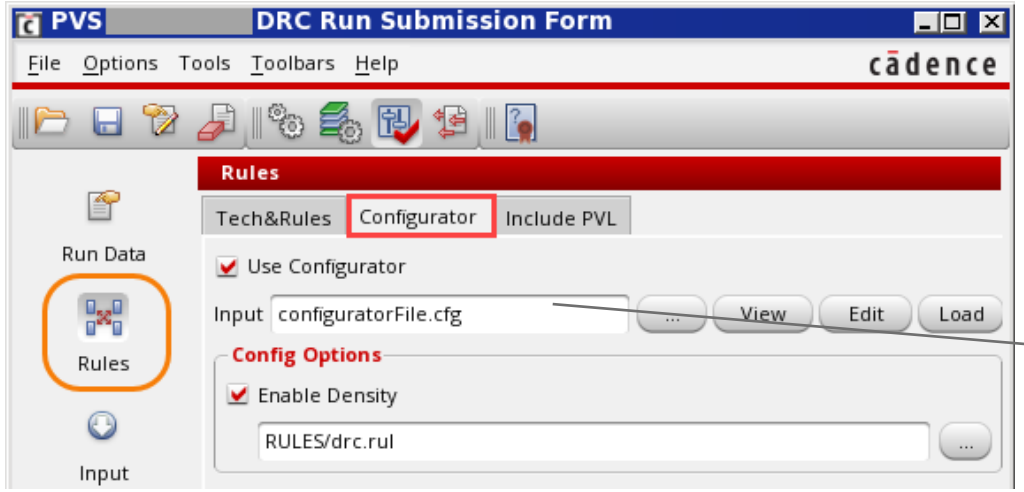
- Add/remove/view/edit custom/foundry rule file(s) to PVS run. (Optional is technology/rule set is specified above.)
- Standalone or append to the Technology Rule Set.
- Multiple rule decks are supported for a DRC run.

DRC Run Submission Form

Rules – PVS Configurator

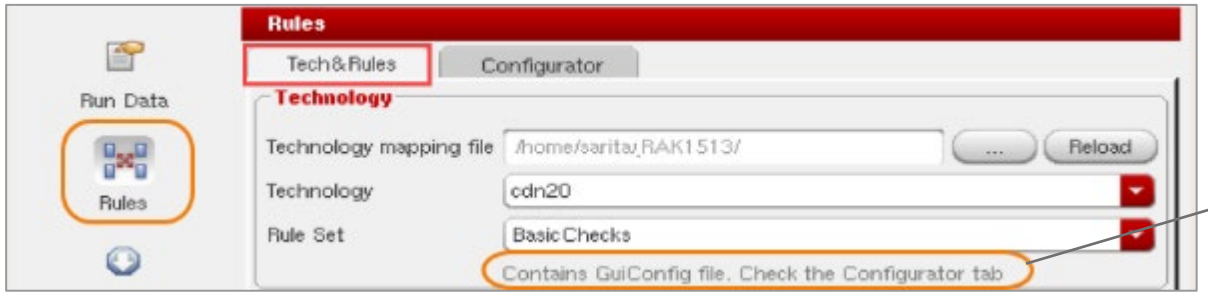
PVS DRC Run Submission Form → Rules → Use Configurator

PVS Configurator is a mechanism to customize the PVS GUI, to allow the selection of items that you can customize in a set of rules.




```
#Example: Configurator file - Enable Density
configoption -type boolean -option DENSITY -text
"Enable Density"
# include Rules File
configoption -type file -default "RULES/drc.rul"
-hidden 0
```

The customization requires a configuration file containing instructions in a specific syntax to enable PVS to create a custom configuration GUI.



If Rule Set contains a Configurator entry, then this statement appears below the rule set.

What Is the PVS Configuration File?

The PVS Configurator-based customization is done using a configuration file (customization file) containing instructions in a specific syntax to enable PVS to create a custom configuration GUI.

- Configuration file: The `#define xxx` or `#undefine xxx` statement allows PVS to select a specific set of rules based on an `#ifdef xxx` conditional block.
- Use the configuration file together with main rule file in the Rules tab of the PVS Run Submission form.
- A control file will be created based on the selections made in the Configurator tab.
  - The final rule set is a combination of the control file + rule file(s).
- The PVL commands from the control file will redefine PVL commands with the same name in rule file(s).
- PVS saves the information about the fields in the form into an automatically created presets file and will be used to restore the fields, including the Configurator tab, during a rerun.

How to Create a Configuration File?

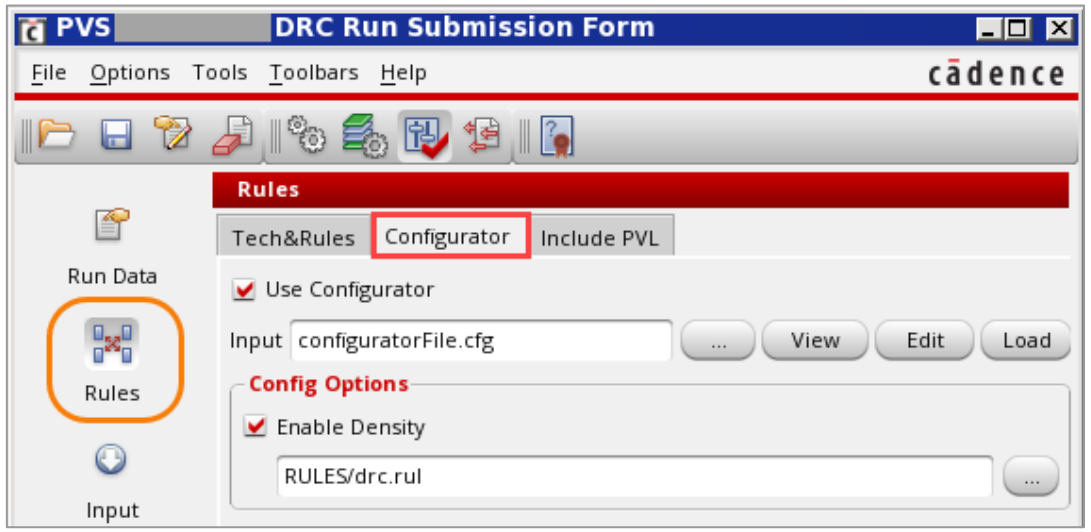
Each entry in the configuration file starts with the word *configoption* followed by one or more type keywords and their values. Each value indicates how the entry is displayed in the configurator tab. You can use Tcl commands within the Configuration file as well.

1. Open a new file.
2. Start each entry with the word *configoption*.
3. Define each configoption as a specific type of entry using type keywords like label, separator, cyclic, Boolean, text, file and action.
4. Use other optional field types like active, justify, option, value, default, hidden, parameters, master etc., customize the configurator GUI.
5. Define environment variable values.
6. Use optional TCL commands to configure the GUI (E.g., Set default value for types, define master/slave fields, run external shell scripts etc.).
7. Save and source the file to see the customized GUI. Set up the GUI and submit the PVS run.

How to Use the PVS Configurator?

PVS Configurator helps customize the rule sets in a PVS GUI with the help of a *Configuration File*.

The configurator can be applied on any PVS job → more flexibility for designers to customize final rule set.



1. Enter the rule file(s) in the Tech&Rules tab.
2. Go to the Configurator tab → Input field.
3. Select the Use Configurator checkbox.
4. Browse and enter the Configuration file.
5. PVS redraws the form with all customizable items.
6. Make choices to customize the current run.
7. Complete the rest of the form.
8. Click Apply or Submit to start the PVS job.

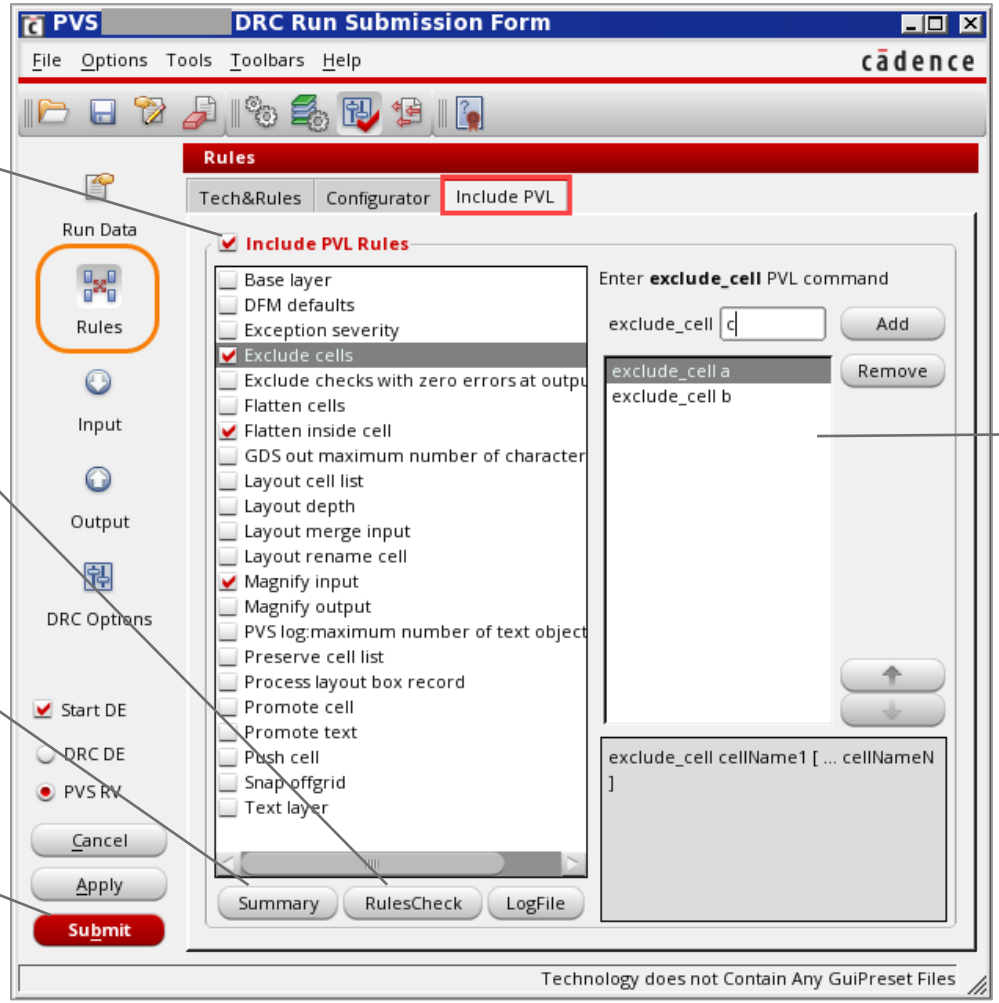
DRC Run Submission Form

Rules – Include PVL

To add additional rules to be passed to PVS run:

- PVS DRC Run Submission Form → Rules → Include PVL

- Select the **Include PVL Rules** checkbox to activate the listed options.
- To run the rule parser to see whether the combined deck (main deck + deck from PVS Include tab) has been parsed by PVS.
- The Summary button shows (on RHS) the final additional rules which will be passed to PVS.
- When the run is submitted, the rules defined here are added to the control file.



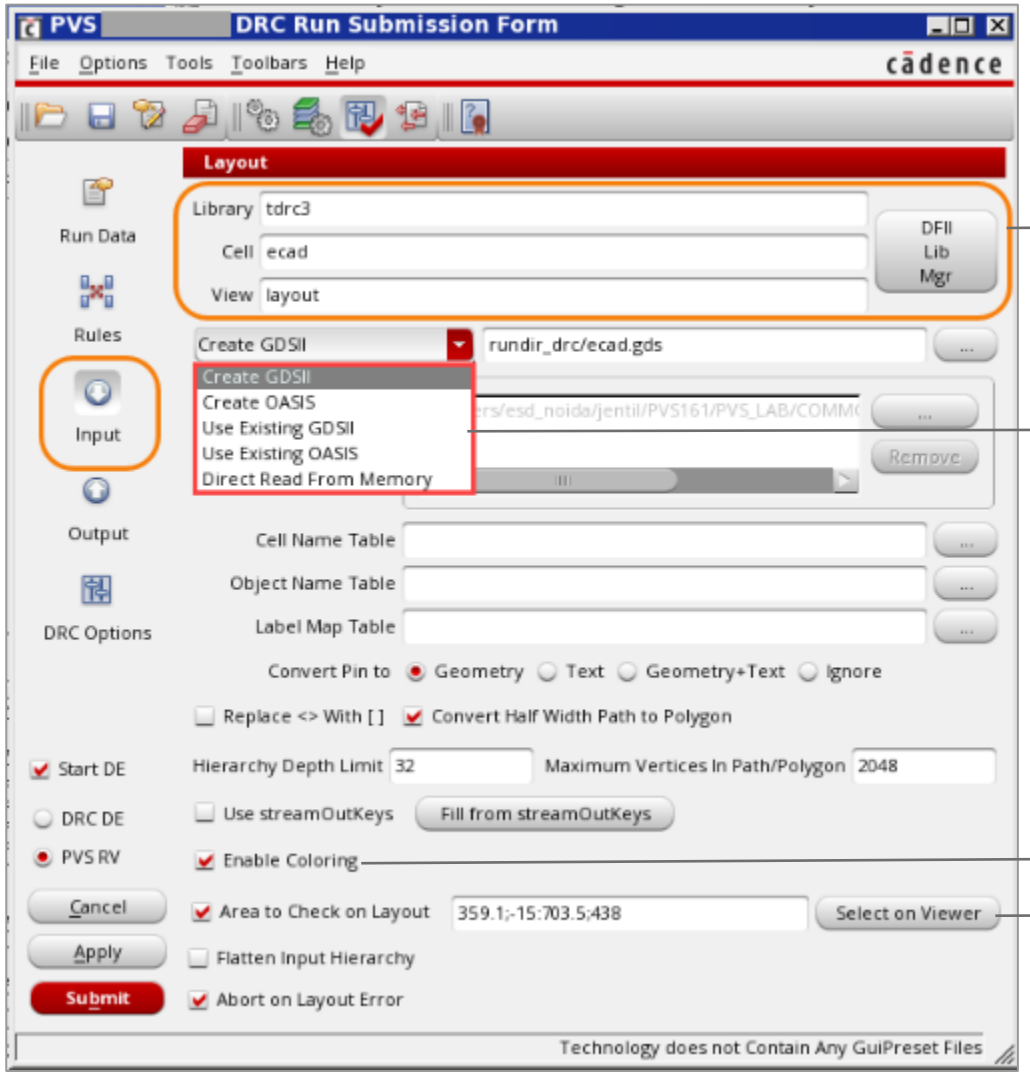
To add rules

- Select the checkbox (on LHS) for the rule that you want to include.
- Fields related to the rule's inputs are displayed on RHS.
- Enter/select/add the required options.

DRC Run Submission Form

Input

PVS DRC Run Submission Form → Input → Layout



Layout: Cellview information
- Library, Cell and View.
- Select **DFII Lib Mgr** to browse to layout cellview.

Layout Input Format options:
- Create GDSII file
- Create OASIS
- Use Existing GDSII
- Use Existing OASIS
- Direct read from Memory

- Enable Coloring – Only for Adv nodes (ICADV 12.3+ ISR releases).
- Virtuoso passes predefined mask data to PVS.

- Area-Based DRC Checking In PVS (details on next slide).

Area-Based DRC Checking In PVS

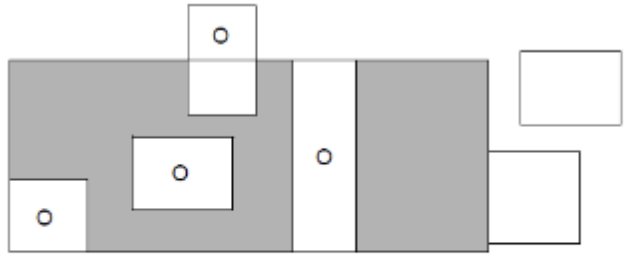
window

```
window x1 y1 x2 y2 [ ... xN yN ]
```

- The window rule selects those database objects that fall within the region defined by the specified x and y coordinates.
  - Allows you to select part of the layout through coordinates and run DRC only in that part of the layout.
  - All shapes to be checked *must* be completely inside the specified window coordinates.
  - E.g., *to verify just the modified area instead of rerunning DRC on the entire design*.
- For the command line::
  - Create a text file with area coordinates (say **add.pvl**):

```
window -10 10 30 30
window 200 200 400 400
```

The coordinates are the corner points of a polygon window.



Database objects marked O are inside the polygon window.

Database objects Polygon window defined by window rule

  - Provide **add.pvl** as a last argument for DRC run.

```
pvs -drc -gds gdsFile -top_cell topcellname <main rule file> add.pvl
```

  - You may also include a coordinates file in your main DRC rules file.

```
include add.pvl
```

DRC Run Submission Form

Output

PVS DRC Run Submission Form → Output

- Report: Name and & limit on number of lines in the report.

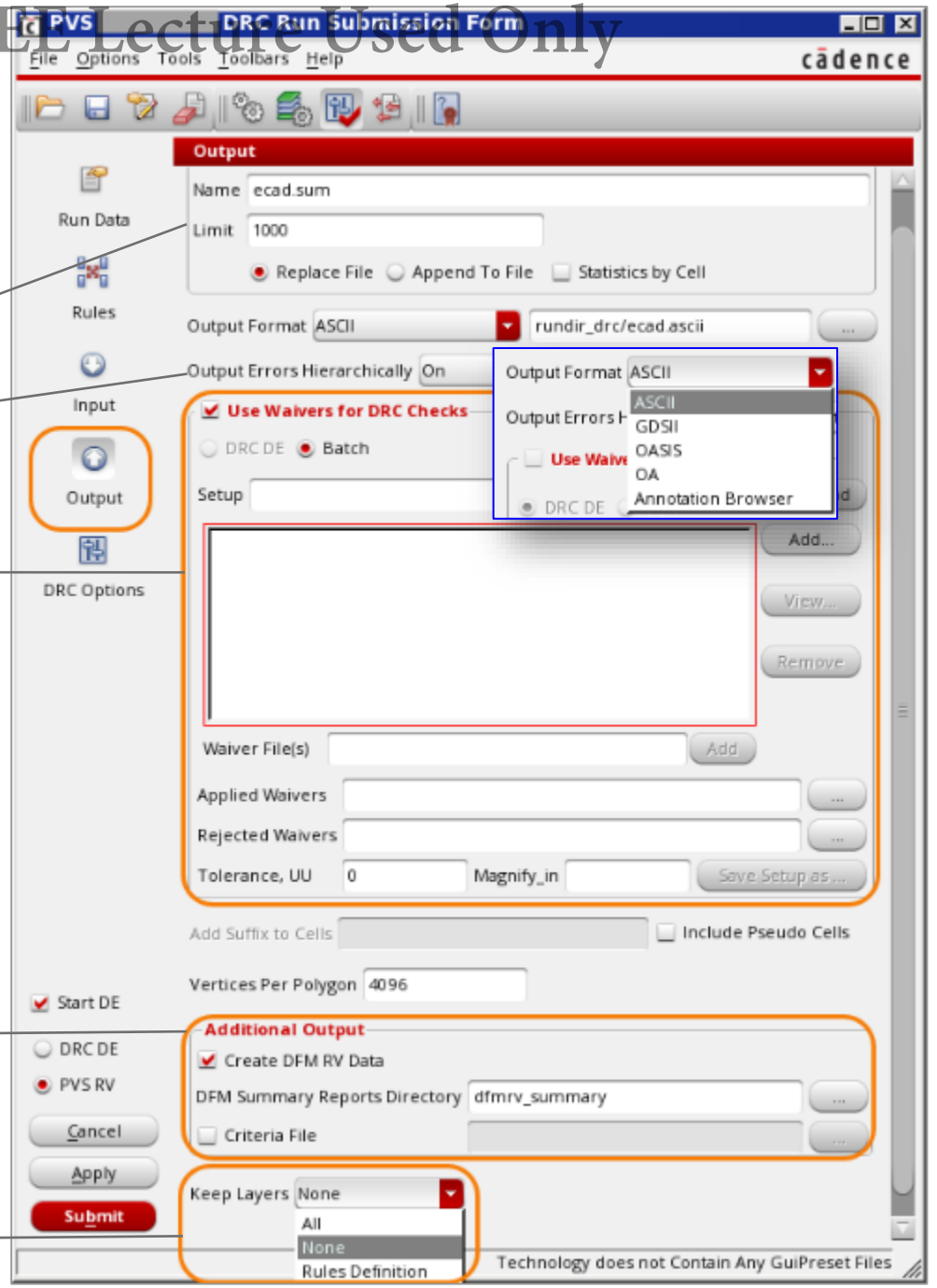
Output Format Control
- ASCII file report / GDSII data file of errors / OASIS file.

PVS DRC Waivers
- Waivers block is disabled by default; activated only when:
  - *Output → Output Errors Hierarchically* is On.
  - *Input → Flatten Input Hierarchy* is deselected.

```
pvs -drc -bwf <waiver_setup_file> drc.rul
```

Additional Output:
- To create data for PVS DFM Results Viewer (RV) GUI.
- View to analyze yield results (PVL rule: dfm_analyze) generated by PVS.

Keep Layers:
- To control visibility of layers (including intermediate) in PVS Layer Viewer.
- 3 options: All, None, Rules Definition.



How to Use PVS DRC Waivers?

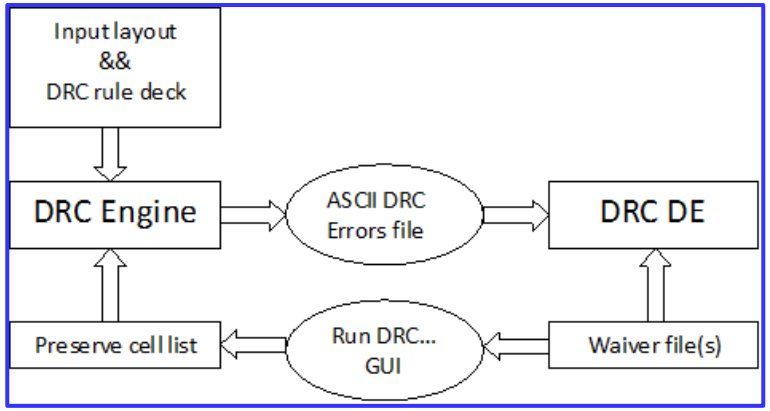
To waive selected violations during PVS DRC run, PVS DRC waivers solution comprises of 2 flows.

A waiver file, with a list of DRCs to be waived, is the key input for both flows.

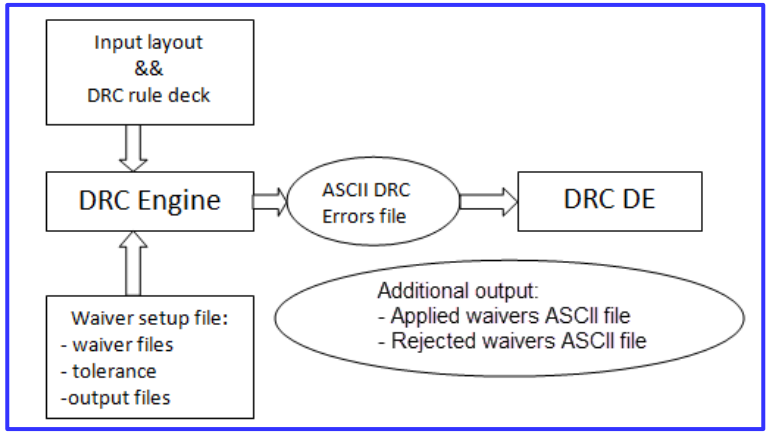
1. DRC DE Based Waiver Flow*:*
   - Post-DRC flow – only from GUI.
   - Preserve cellist required – causes runtime degradation.
   - Waivers are applied on the resultant DB through DRC debug environment.
   - Accessed only from the DRC Run Submission GUI.
   - Waiver file and DRC output must be in ASCII format.
2. Signoff DRC Waiver Flow (Recommended):
   - Pre-DRC flow – Can run from GUI or command line.
   - Waiver file is considered by PVS DRC engine, thereby DRC output file is devoid of waived DRCs.
   - Inputs (waiver file, tolerance etc.) are fed through a waiver setup file.
   - Preserve cellist not required – faster run.
   - Supported DRC run output formats: ASCII/GDSII/OASIS/OA.

```
pvs -drc -bwf <waiver_setup_file> drc.rul
```

DRC DE Based Waiver Flow



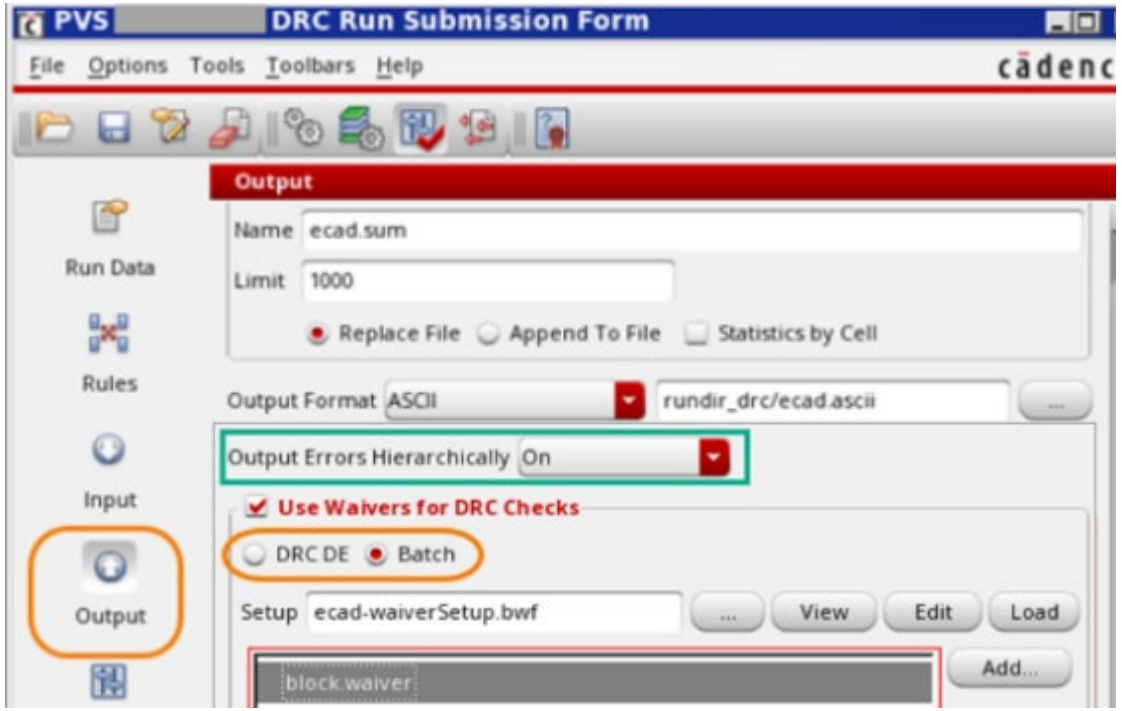
Signoff DRC Waiver Flow



PVS DRC Waivers Setup

PVS DRC Run Submission Form → Output → Use Waivers for DRC Checks → DRC DE or Batch.

- Select DRC DE: To use DRC DE based Waiver flow.
- Select Batch form: To use Sign-off DRC Waiver flow.
- The Waivers block is enabled only if:
  - Output panel: Output Errors Hierarchically option is selected.
  - Input panel: Flatten Input Hierarchy option is deselected.

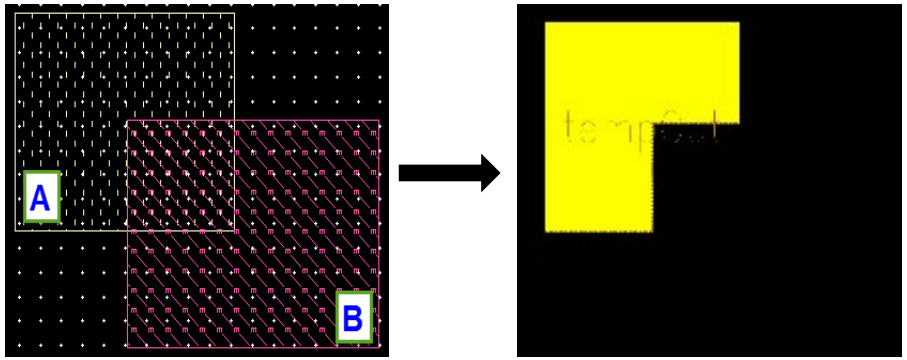


What Is Layer Viewer?

Keep Layers

The Layer Viewer shows the layers processed by a PVS or Pegasus run (including intermediate) based on the Keep Layers selection made in the GUI or the **`keep_layers`** batch command.

- Application – To debug intermediate or derived geometric layers:
  - Helps rule deck writers to debug DRC rules
- License required: Phys_Ver_Sys_Design_Ana

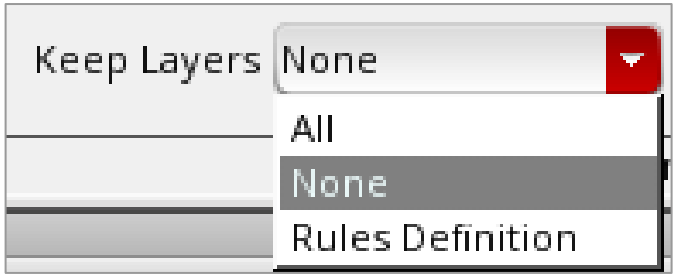


Setup Layer Viewer

DRC Run Submission Form → Output → Keep Layers

Virtuoso Layout Suite → PVS / Pegasus → Layer Viewer

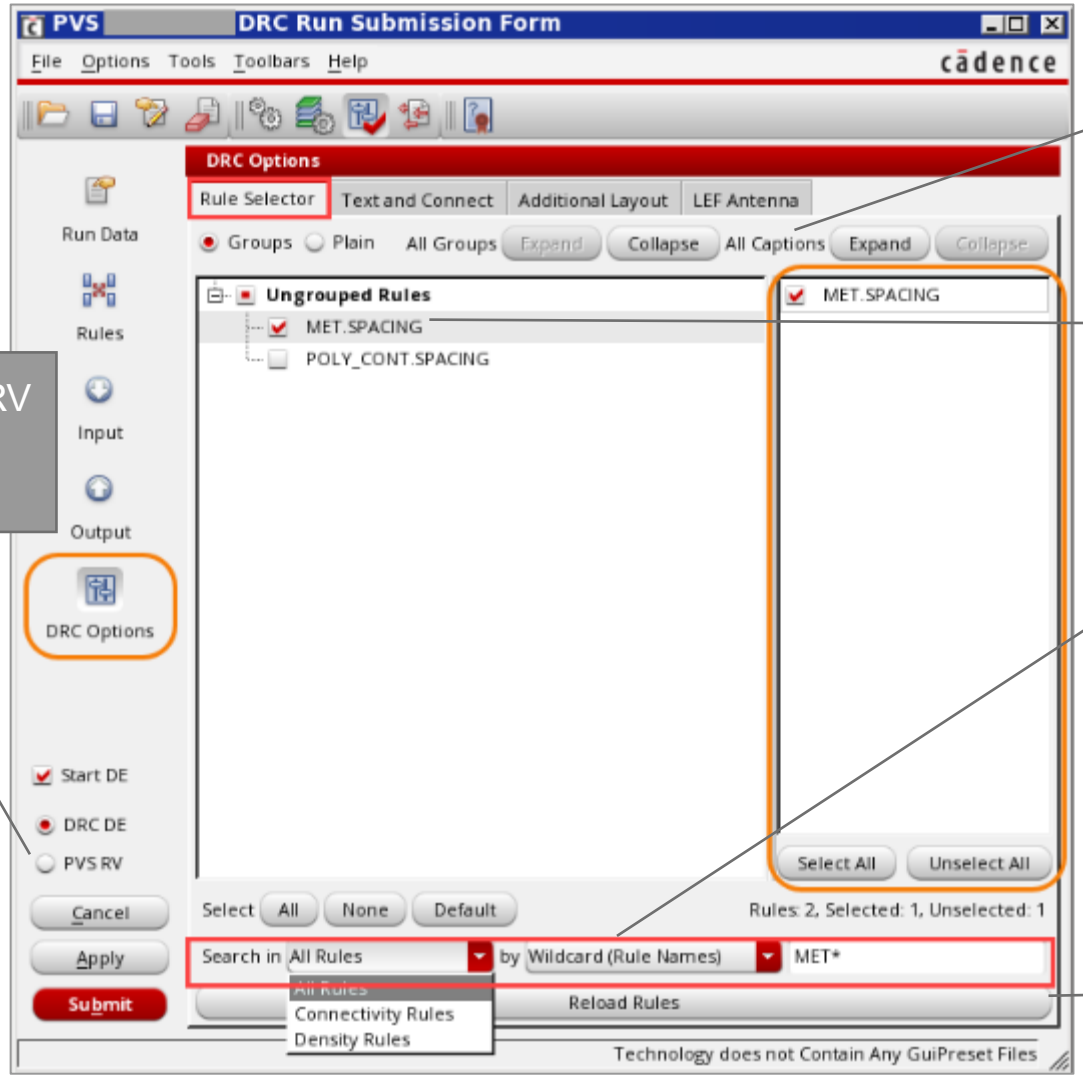
- Visibility of layers (in PVS Layer Viewer) can be controlled using 3 options in the Keep Layers drop-down:
  1. All: Allows you to keep all layers including intermediate.
  2. None: Do not save layers. This is the default.
  3. Rules Definition: Allows you to keep layers on the basis of rules file (**`keep_layers`**).
- In Virtuoso, processed layers are viewed through Layer Viewer invoked from the PVS/Pegasus drop-down.



DRC Run Submission Form

DRC Options

PVS DRC Run Submission Form → DRC Options → Rule Selector



Start DRC DE/PVS RV
- To run PVS Results Viewer.

- List rules in group/plain.
- Expand/collapse the list of rules.

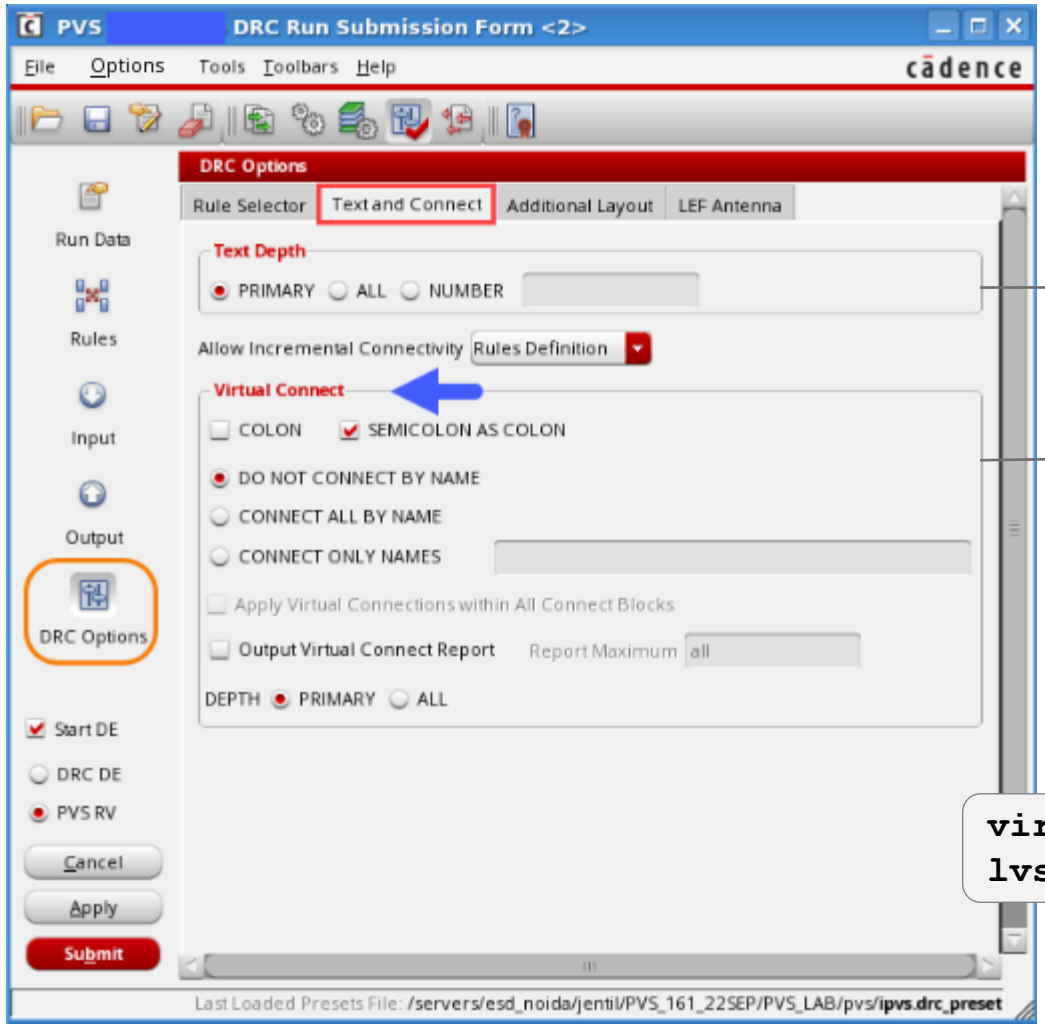
- Select or deselect rule sets or rule groups previously added in the Rules form.

Customize your search results using various drop-down options:
- All Rules, Connectivity Rules and Density Rules.
- Wildcard options: (Rule Names, Rule Captions).
- Reg. Exp. options: (Rule Names, Rule Captions).

Reload Rules
- Changes made to the rules previously loaded will be refreshed by clicking the **Reload Rules** button.

Text Depth and Virtual Connect

PVS DRC Run Submission Form → DRC Options → Text and Connect



Text Depth:

- PRIMARY: Top level only. Default.
- ALL levels.
- NUMBER: Specific level only.

Virtual text connections:

- Instruct LVS tool to connect nets ONLY for verification runs. No physical connections.
- Colon connect: Texts are named with a ":" or ";"
- Additional options to:
  - Generate a report.
  - Define Depth.
- **Note**: Remove virtual connect definitions before signoff LVS run to avoid masking actual opens.

```
virtual_connect -colon yes -depth 3 -name VDD VCC
lvs_join_nets cellA VDD VCC vdc -layout
```

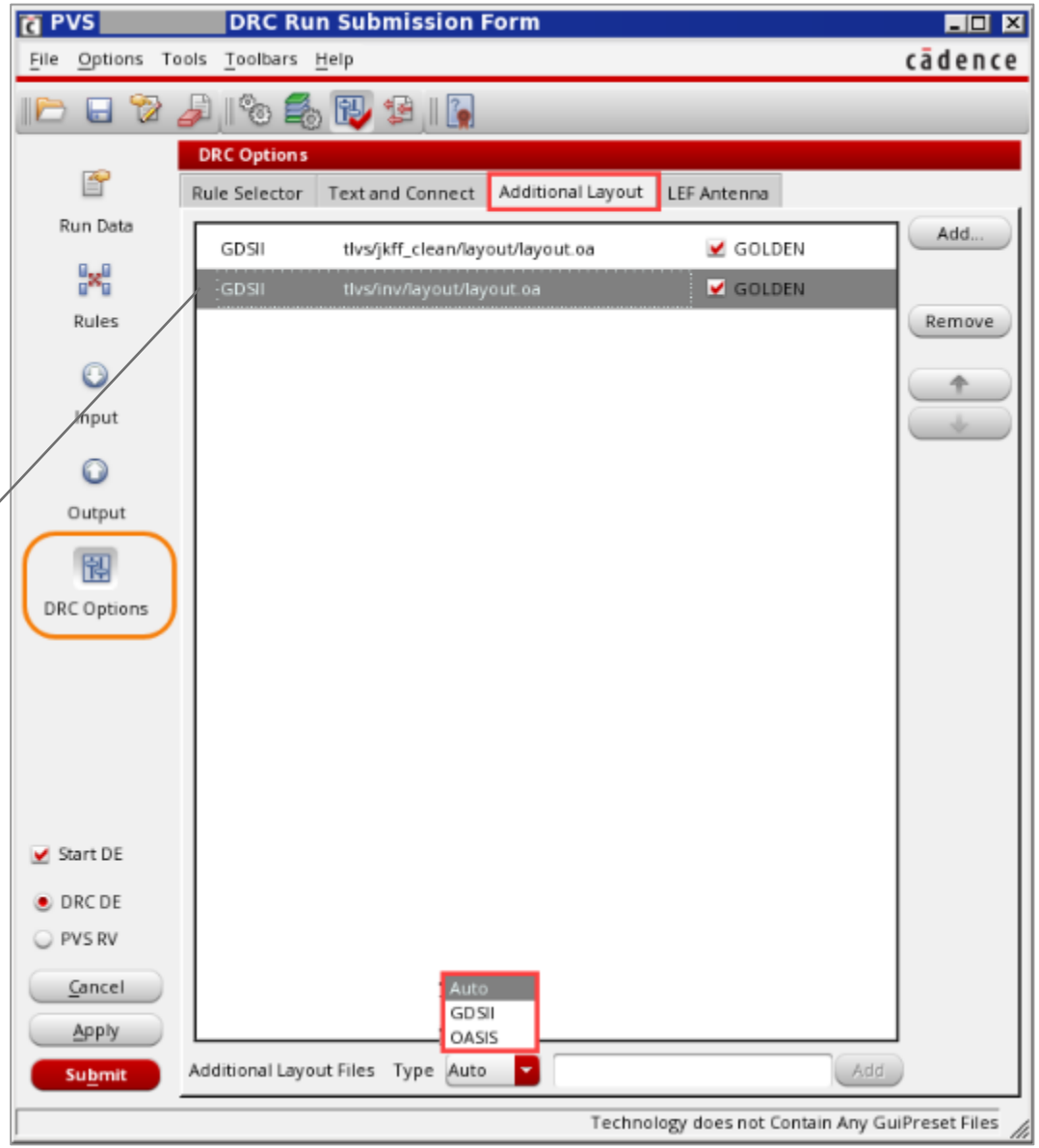
PVS DRC GOLDEN Functionality

PVS DRC Run Submission Form → DRC Options → Additional Layout

- GOLDEN functionality: To treat certain cells as golden for signoff flow. Skips error reporting on listed layers or cells.
- Waiver-like flow for signoff or a do not modify flow for designers.

You can specify the layout(s) to be considered as GOLDEN (use checkbox).
- Selected layouts don't merge with primary layout.
- Supports GDSII/OASIS.
- Wildcard is supported in specifying file name(s)

Recommendation: Do not use the GOLDEN feature with VIPVS Post-Edit mode and PVS window mode to avoid false error reporting.

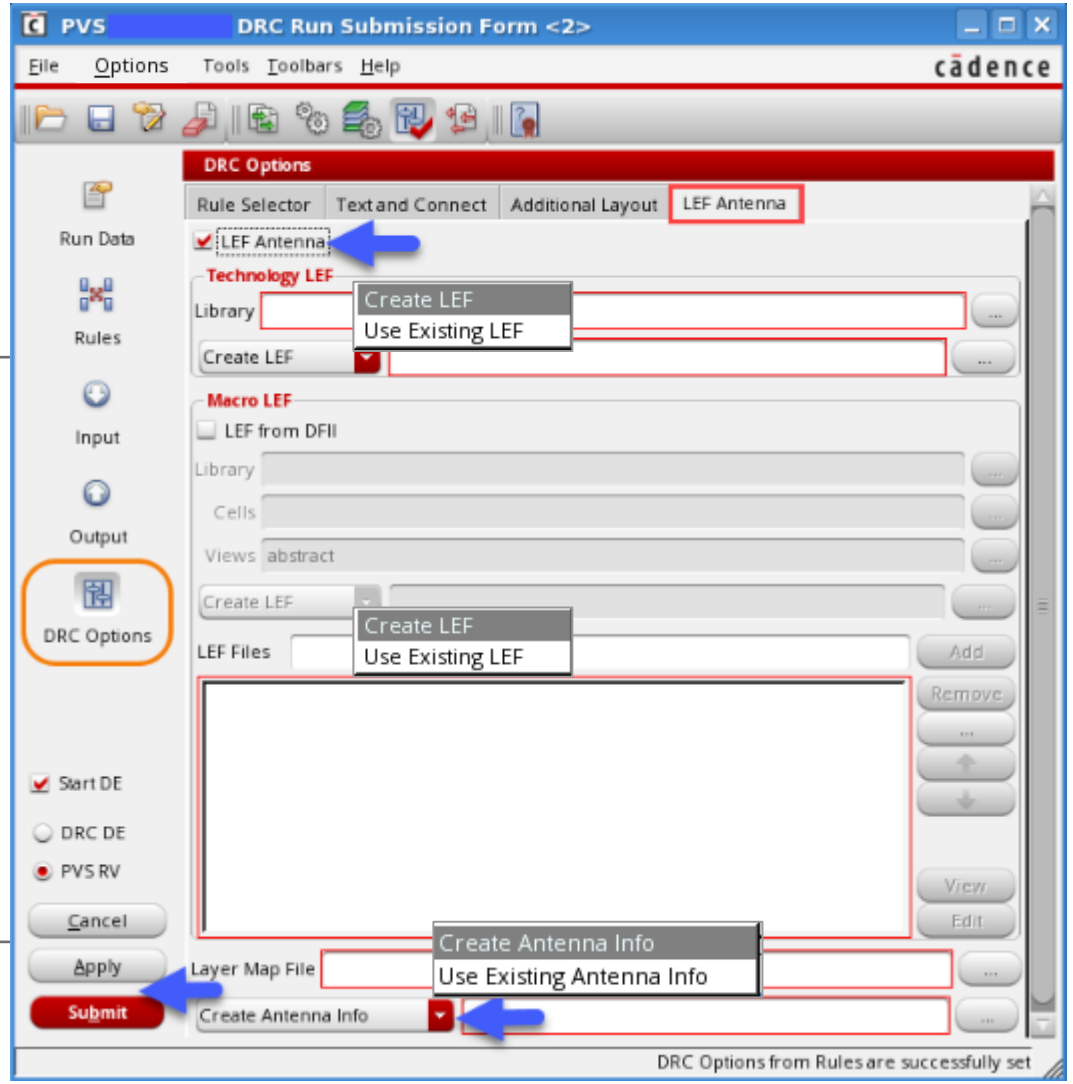


LEF Antenna

PVS DRC Run Submission Form → DRC Options → LEF Antenna

1. You can specify LEF Antenna Pin info to be considered during the run.
2. You may create a new LEF or use the existing LEF (from foundry or pre-generated).

- PVS uses the *genAntennaInfo* script to create a file with Antenna process info for IP cells:
- Input for Antenna checking: areas, perimeters or shape counts.
- Condition: The LEF FILE, Layer Map File and PVL RULE DECK must be consistent with each other.



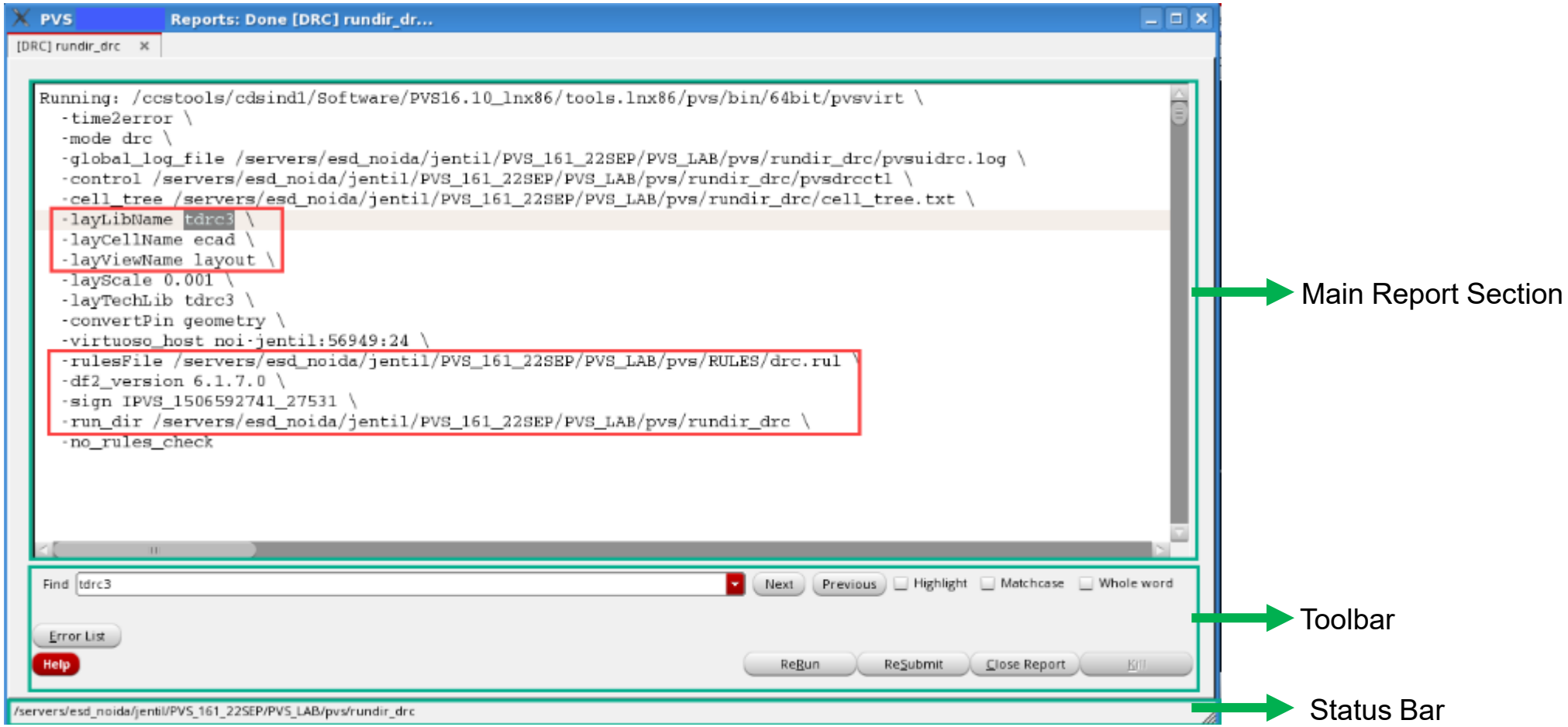
Note: You are all set to start the DRC run; click **Apply** or **Submit**.

PVS DRC Debugging

Reports Window

PVS DRC Run Submission Form → Submit → PVS Reports

The PVS Reports window comes up when you submit the PVS run. The window displays the summary of inputs to help debug startup issues.



What Is the PVS Run Command File?

run.pvs

A shell script command file *run.pvs* is created in the run directory after a PVS run is completed. The command file contains the command line to run PVS. You can use this file to rerun PVS in batch mode after it is run in the GUI mode.

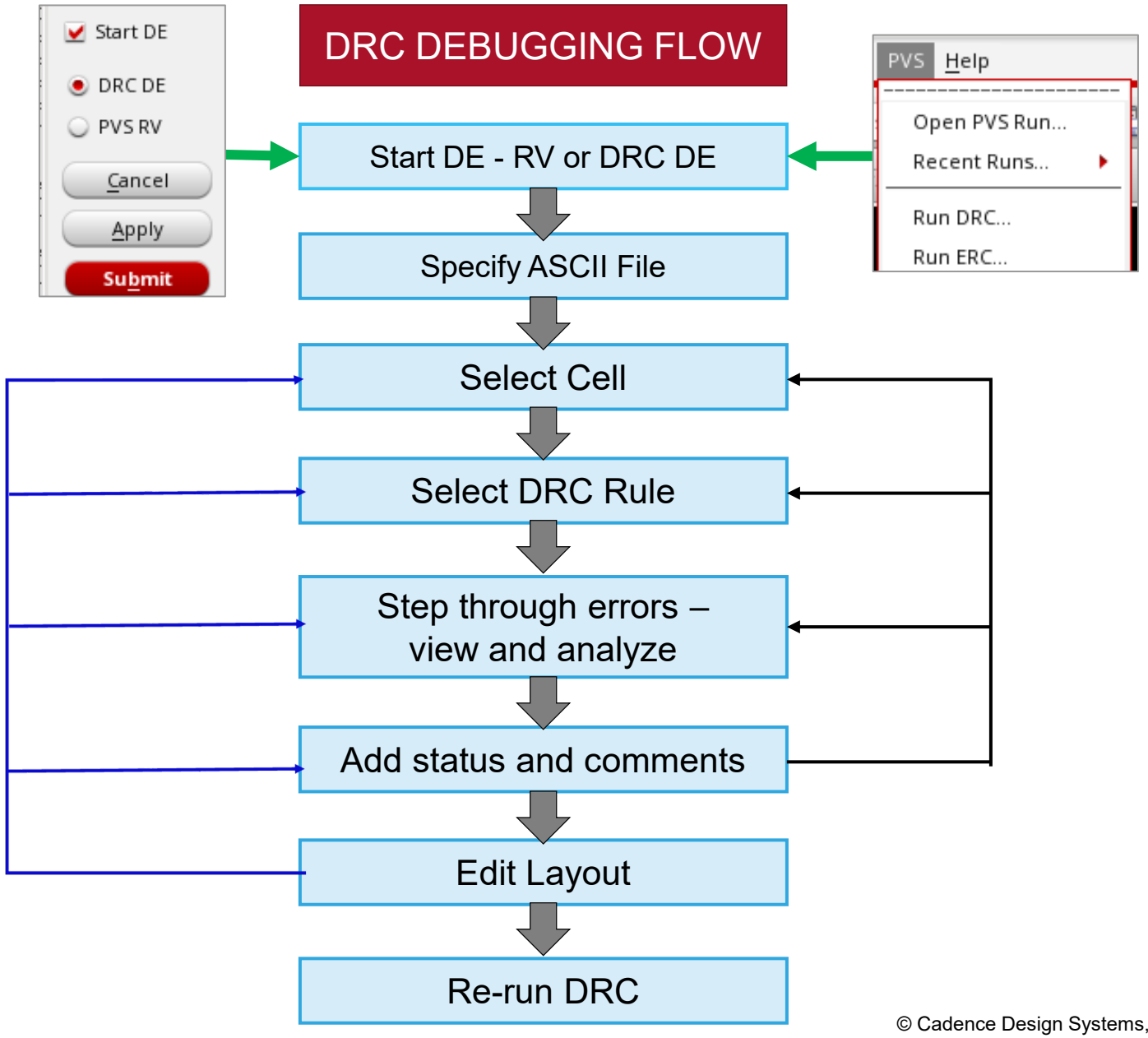
- Pre- and post-triggers and environment variables setting from configurator, if any, are captured.
- Control file, gds file etc., are used in PVS run as is.
- Layout library/location details from the disk will be captured in the script.

++++ Example of run.pvs Script with template ++++

```
#!/bin/sh -f
###### This run file is generated by PVS UI
###### PVS Version 18.20-pxxx
export VARIABLE_1="VALUE1"
export VARIABLE_2="VALUE2"
./batch_pre.trg parameter1 parameter2 ..
pwd_d=`pwd`
cd <path to rundir>;\
<full path>/pvs \
          -drc \
          -top_cell TOP_CELL \
          -ui_data \
          -control <run_dir>/pvsdrcctl \
          -cell_tree <run_dir>/cell_tree.txt \
          <run_dir>/.config.rul\
          <full_path>/drc.pvl
cd $pwd_d
./batch_post.trg parameter1 parameter2 ...
```

- Header of script
- Environment variables from configurator, if any
- Execution of batch pre-trigger script, if any
- Save current directory and cd to PVS runDir
- Command line generated by PVS GUI
- Return to CWD and run post-trigger script, if any

Flowchart: DRC Debugging Flow



What Is the PVS Results Viewer (RV)?

Results Viewer is the single entry point to all results viewing and management tools in PVS.

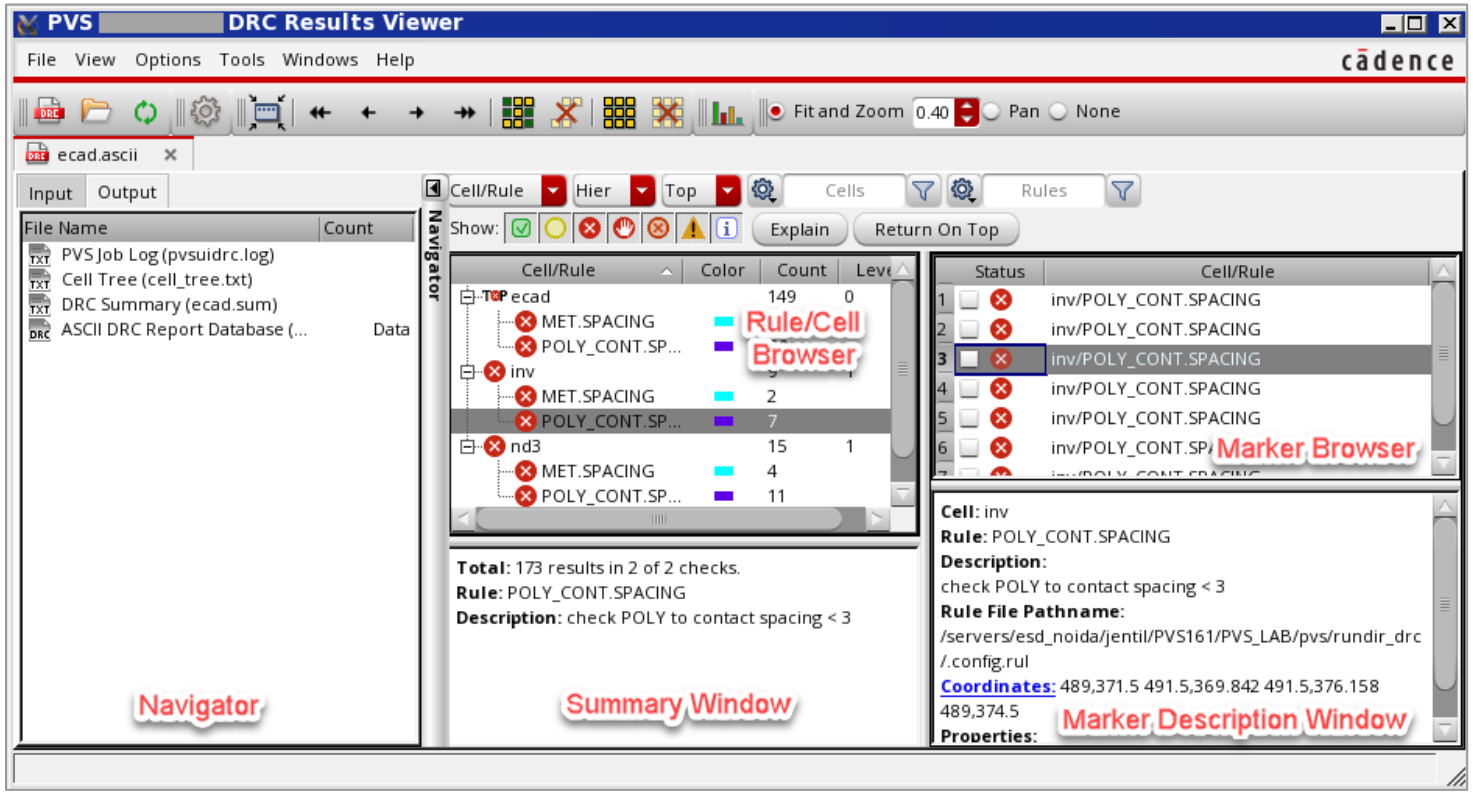
Concept: One run directory – One application instance.

- To view and analyze DRC, Antenna, Density, ERC violations
  - From Virtuoso, Innovus, PVS QuickView platforms & VIPVS
- Results viewer acts as a:
  - Directory browsing tool
  - Launch tool for legacy browsers (*lvsbrowser*, *drcbrowser*)
  - Unified platform for integration of various results management utilities (implemented as statically linked libraries)
- Results Viewer GUI utilities:
  - Status bar, Toolbar and Menu management
  - Keyboard shortcuts editor for *ActionManager*
  - Single Preferences window

PVS DRC Results Viewer

Features

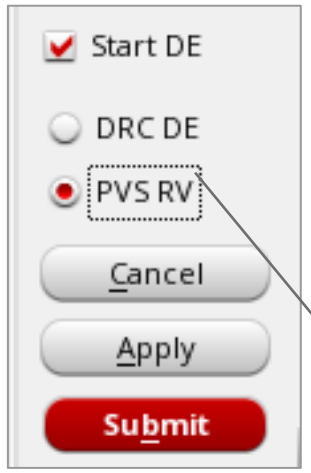
- PVS RV features list
  - Histograms
  - Heatmap (colormap) for density
  - Compact mode
  - Open multiple files at different tabs
  - Assign colors for different rules
  - File Browser
  - Multiple layout windows support in Virtuoso
  - Hyperlink to DRM
  - Merge result from different rules or files



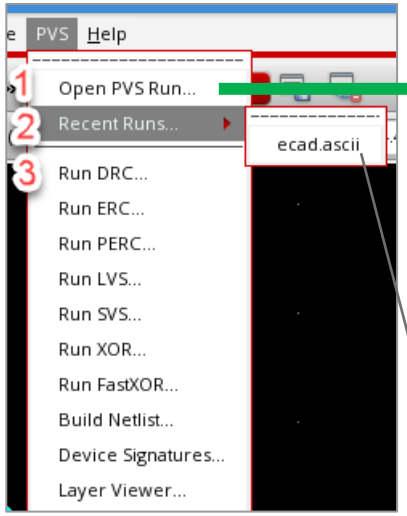
How to Invoke the PVS Results Viewer?

Virtuoso → PVS → Recent Runs
Virtuoso → PVS → Open PVS Run
Virtuoso → PVS → Run DRC → PVS DRC Run Form → Start DE → PVS RV

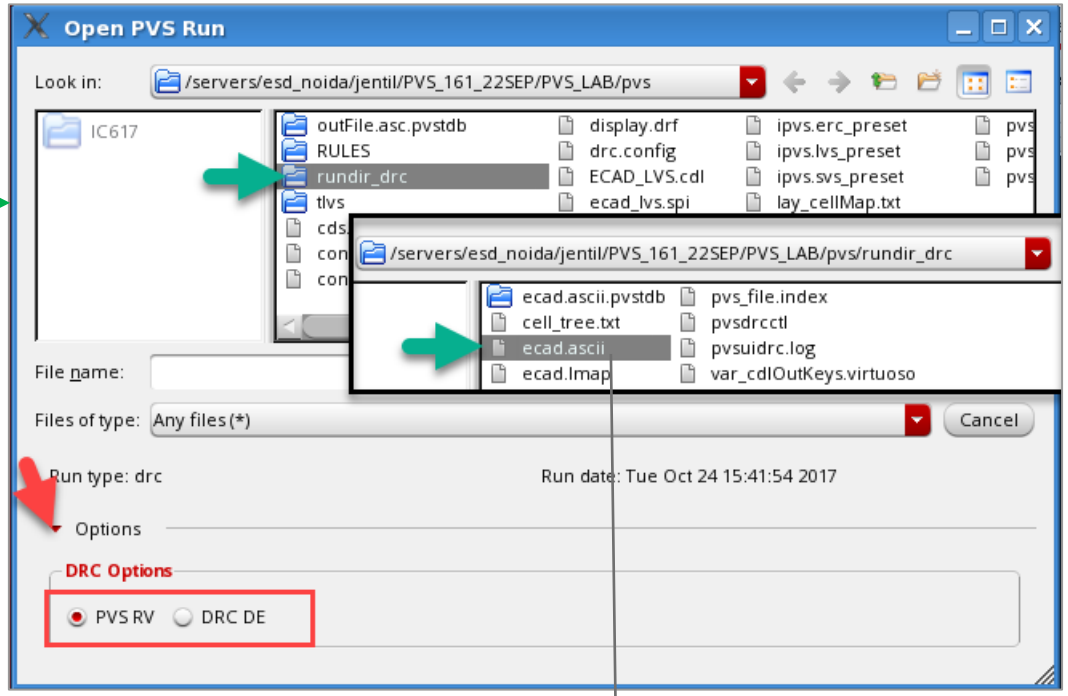
- Skill function: **`pvsStartRV`**



- PVS → Run DRC → PVS DRC Run Submission Form → Start DE → PVS RV (Results Viewer)



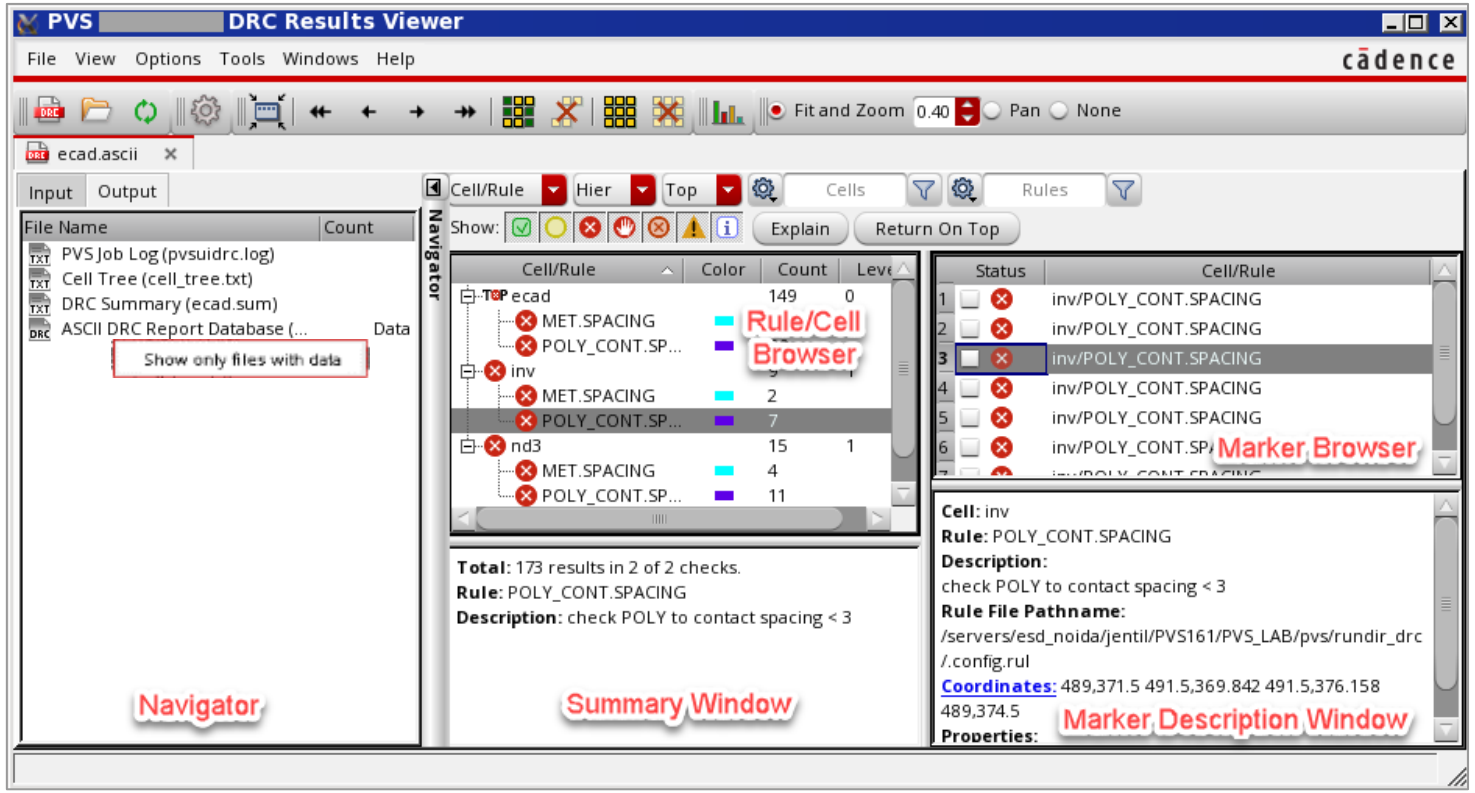
- PVS → Recent Runs
- From the list of recently performed DRC runs



- PVS → Open Run → Open PVS Run form → Select ASCII file (or directory) → Open → PVS RV
- Supports both Run Directory & ASCII file
- Set DRC options to PVS RV

Overview of the PVS Results Viewer Browser

Let's look at an overview of the major windows in the PVS Results Viewer.



5 windows on PVS RV to debug DRC

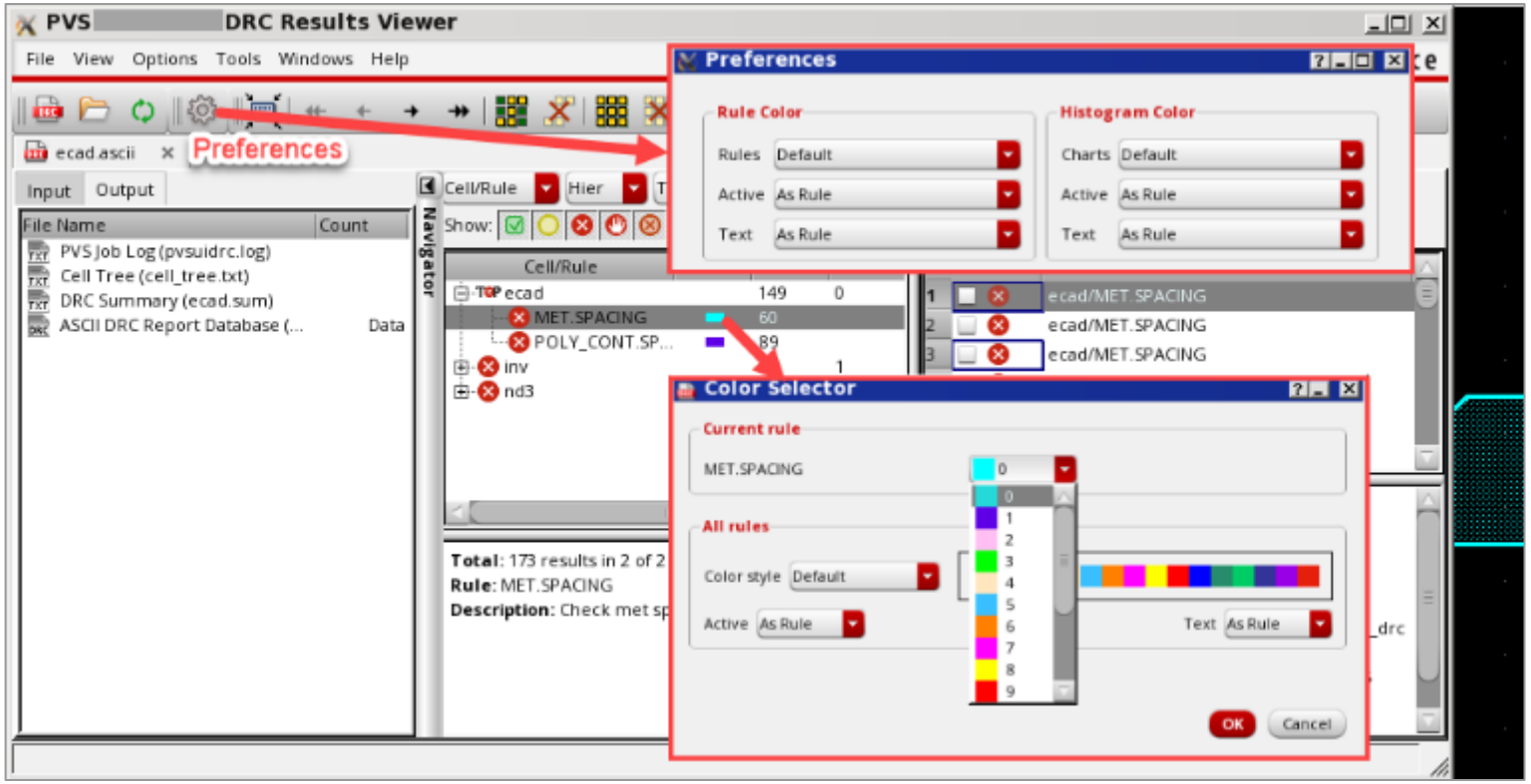
- Navigator
  - Can be minimized
  - Lists Input/Output Files – open files in new tabs
- Rule/Cell Browser → Highlight/View violations:
  - Cell/Rule or Rule/Cell mode
  - Rule Color selection
  - List rules based on Count, Level, Status etc.
  - Hierarchy / Flat & Top / Cell / EIP / Filtered mode
- Summary Window: Summary of Rules
- Marker Browser
  - List/sort selected rules in Rule/Cell browser
  - Highlight selected error markers in Layout
  - Checkbox-based marker highlight controls
  - View-By-Area option
  - Status of Rules can be changed and listed
- Marker Description Window: Details of selected marker

How to Use the PVS DRC Results Viewer?

Here is a step-by-step procedure on using multiple features of the PVS DRC Results Viewer. These features are spread across various browsers in PVS Results Viewer – namely Navigator, Rule/Cell Browser, Summary window, Marker Browser & Marker Description window.

Step 1: Assign / Change Colors for Rules

Assign / Change Colors for Rules → Invoke Compact mode of PVS RV → Debug Density Violations → Hyperlink Rules to DRM

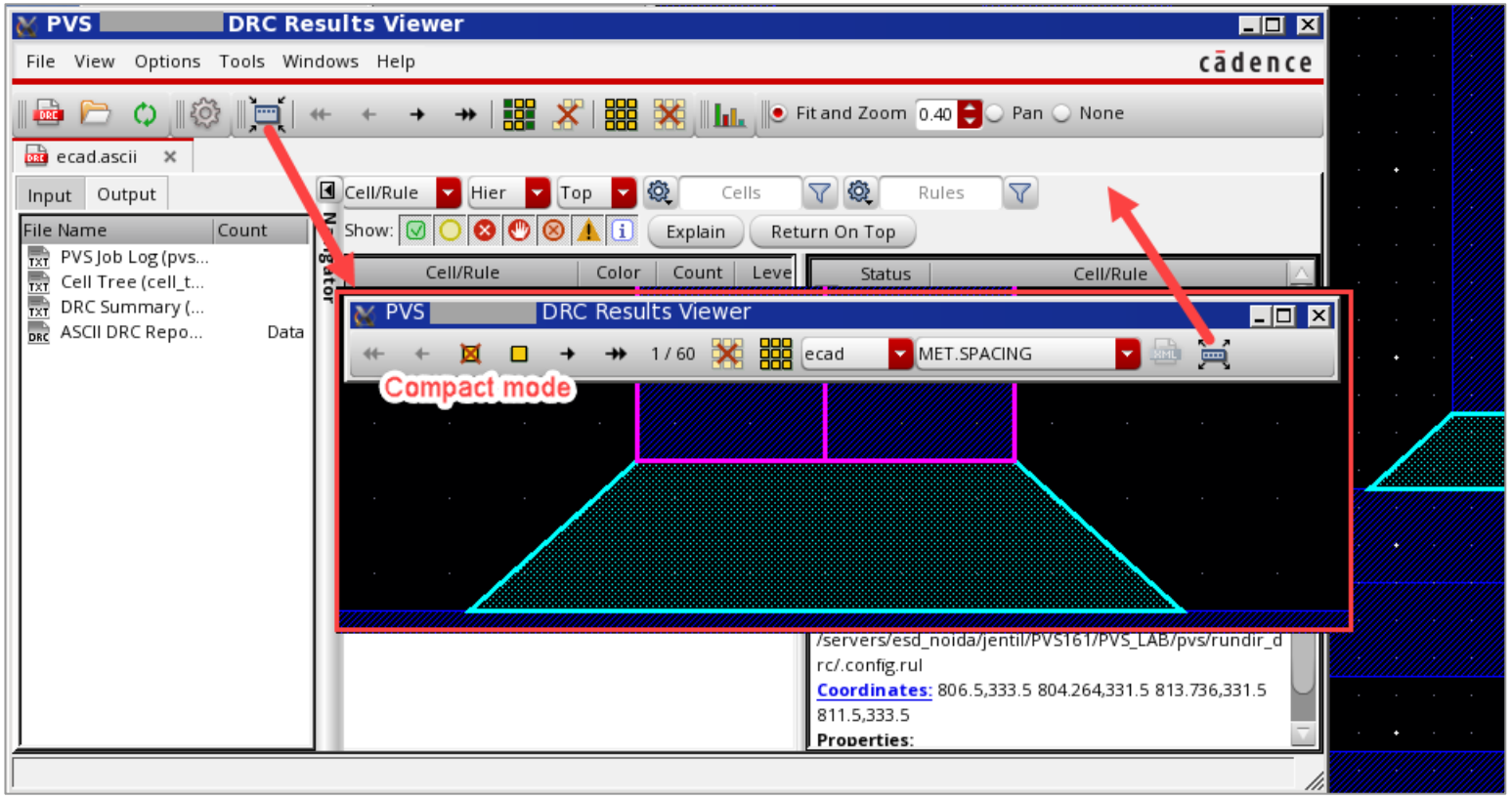


How to Assign/Change colors for rules?

- Color related changes can be done from:
  - *Color Selector*
  - *Preferences Form.*
- Right-click on the color in the Rule/Cell Browser to invoke *Color Selector*.
- Multiple color schemas are available to chose from.
  - Default: *16_colors*
- *Active* stands for the color of currently active marker, of the selected rule, in the *Marker Browser*.
  - By default, it is set as same *As Rule*.
- Color of the *Text* on the markers.
  - Separate color or same *As Rule*?

Step 2: Invoke Compact Mode of PVS RV

Assign / Change Colors for Rules → **Invoke Compact mode of PVS RV** → Debug Density Violations → Hyperlink Rules to DRM

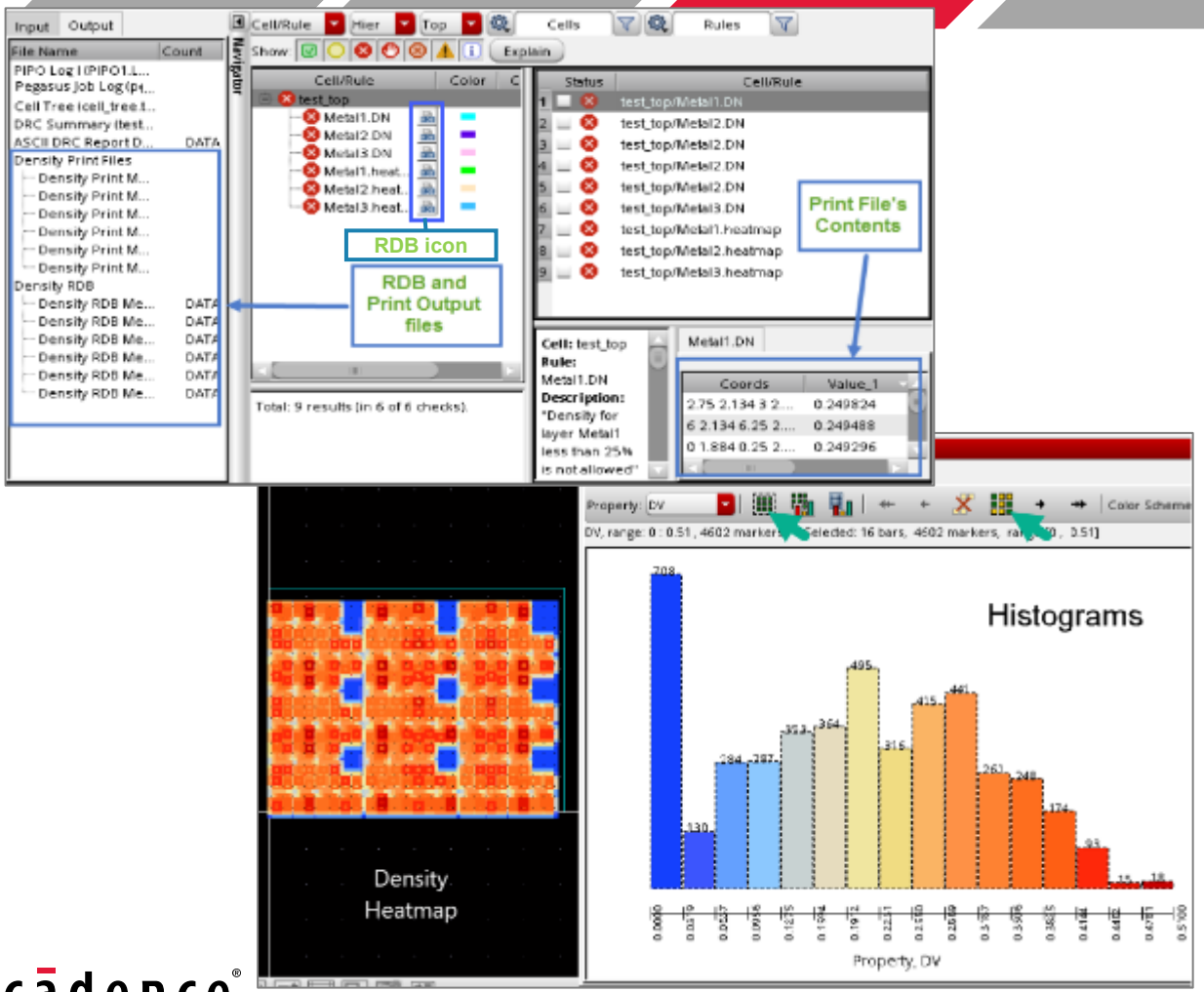


How to invoke compact mode in the PVS Results Viewer?

1. Click the compact mode icon on the PVS RV toolbar or View menu.
2. Shows PVS RV as compact error marker navigation bar.
3. Compact mode saves space on screen while catering basic capabilities → navigation, highlighting, filtering.
4. Click the expand icon on the right end of the GUI to return back to full RV mode.

Step 3: Debug Density Violations

Assign / Change Colors for Rules → Invoke Compact mode of PVS RV → **Debug Density Violations** → Hyperlink Rules to DRM

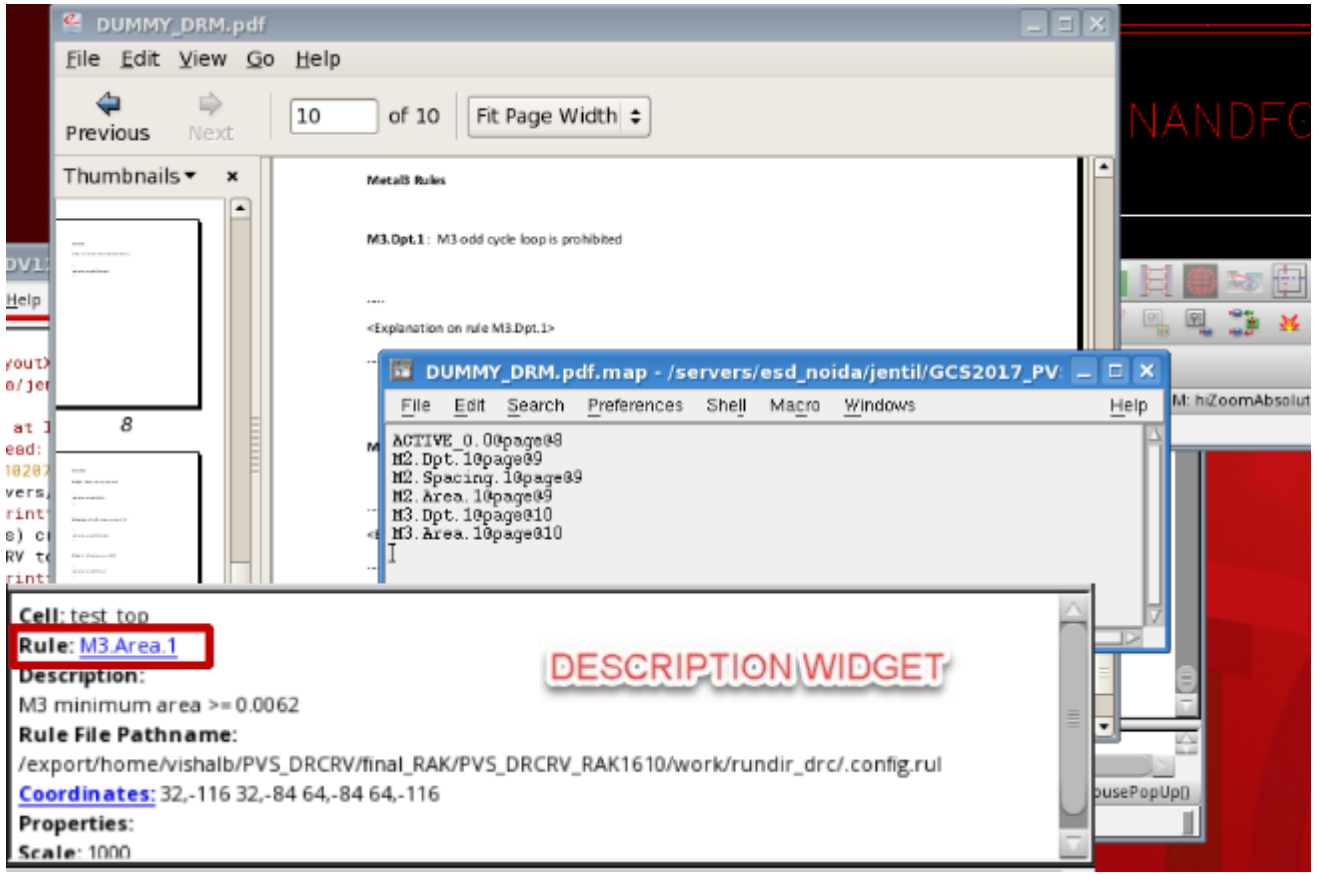


How to debug Density Violations with the PVS Results Viewer?

- Density rules are listed in rule/cell browser:
  - Selected markers gets highlighted in layout.
  - Click the rule or icon to open it in new tab.
  - Density rules usually generate two additional files – PRINT and RDB files.
    - PRINT files contain density window coordinates.
    - RDB files indicated by an icon next to rule.
  - Histograms: Created based on:
    - RDB properties (DV, DA).
    - Marker selection, Layout Area, SQL Query.
  - Density heat-mapping:
    - Highlight histogram selections onto Layout.
    - To analyze problematic density areas.

Step 4: Hyperlink Rules to DRM

Assign / Change Colors for Rules → Invoke Compact mode of PVS RV → Debug Density Violations → Hyperlink Rules to DRM



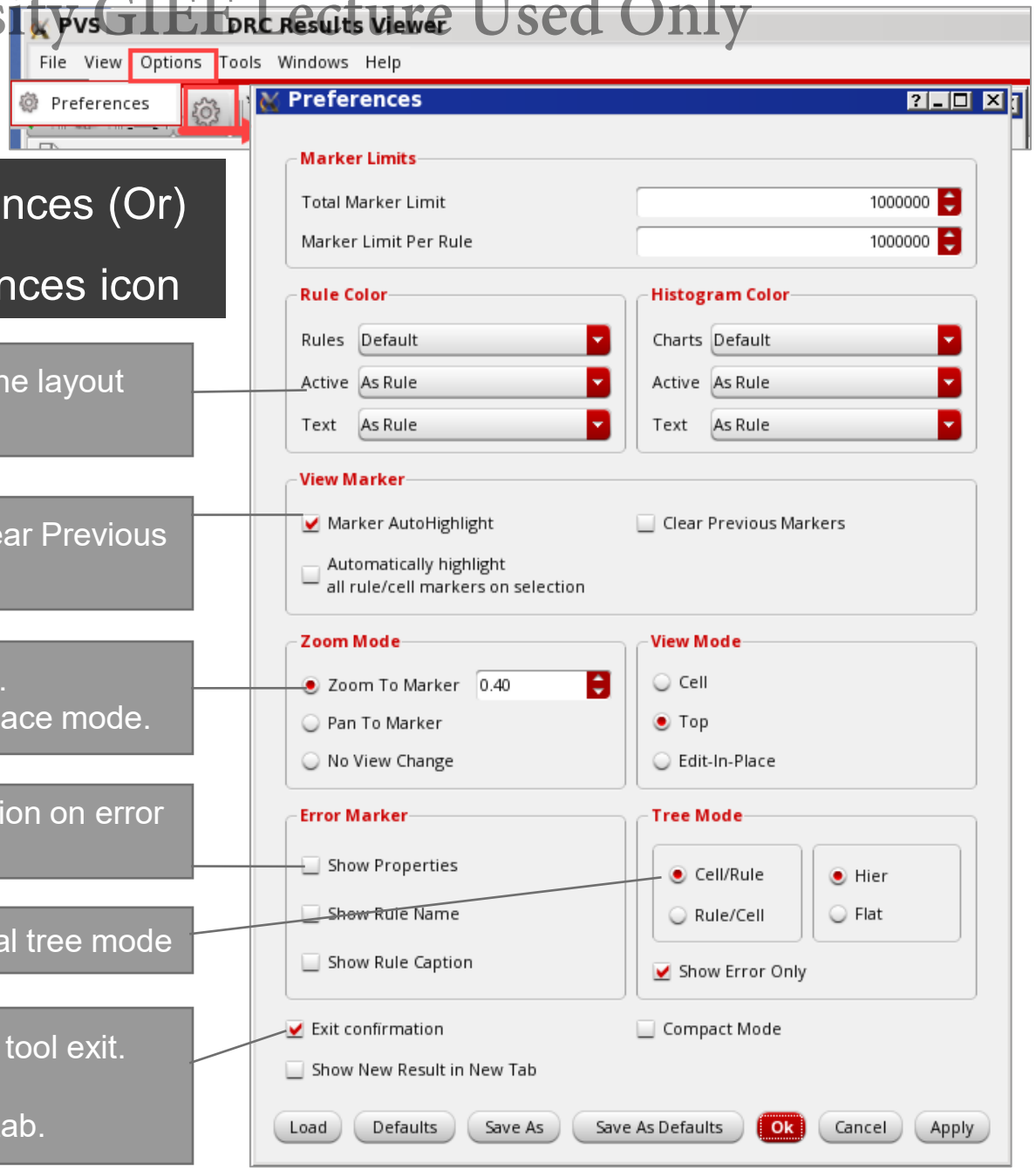
How to hyperlink DRC rules to corresponding DRM(Design Rule Manual) pages?

1. Keep the following files ready.
   - DRC rule to DRM mapping file (format: ***RuleName>@<action>@<token>***)
   - DRM file
2. Hyperlinking via PVS RV GUI: RMB on marker Browser to bring context menu. In the menu, map the following files:
   - DRM → Select DRM File
   - DRM → Select DRM Map File
3. Hyperlinking via environment variables:
   - setenv PV_DE_DRM_FILE <path>
   - setenv PV_DE_DRM_MAP <path>
4. That turns rule names in the description section to a hyperlink. Click the link to open the DRM PDF file page.

PVS Results Viewer Preferences

PVS Results Viewer → Options → Preferences (Or)

PVS Results Viewer → Toolbar → Preferences icon

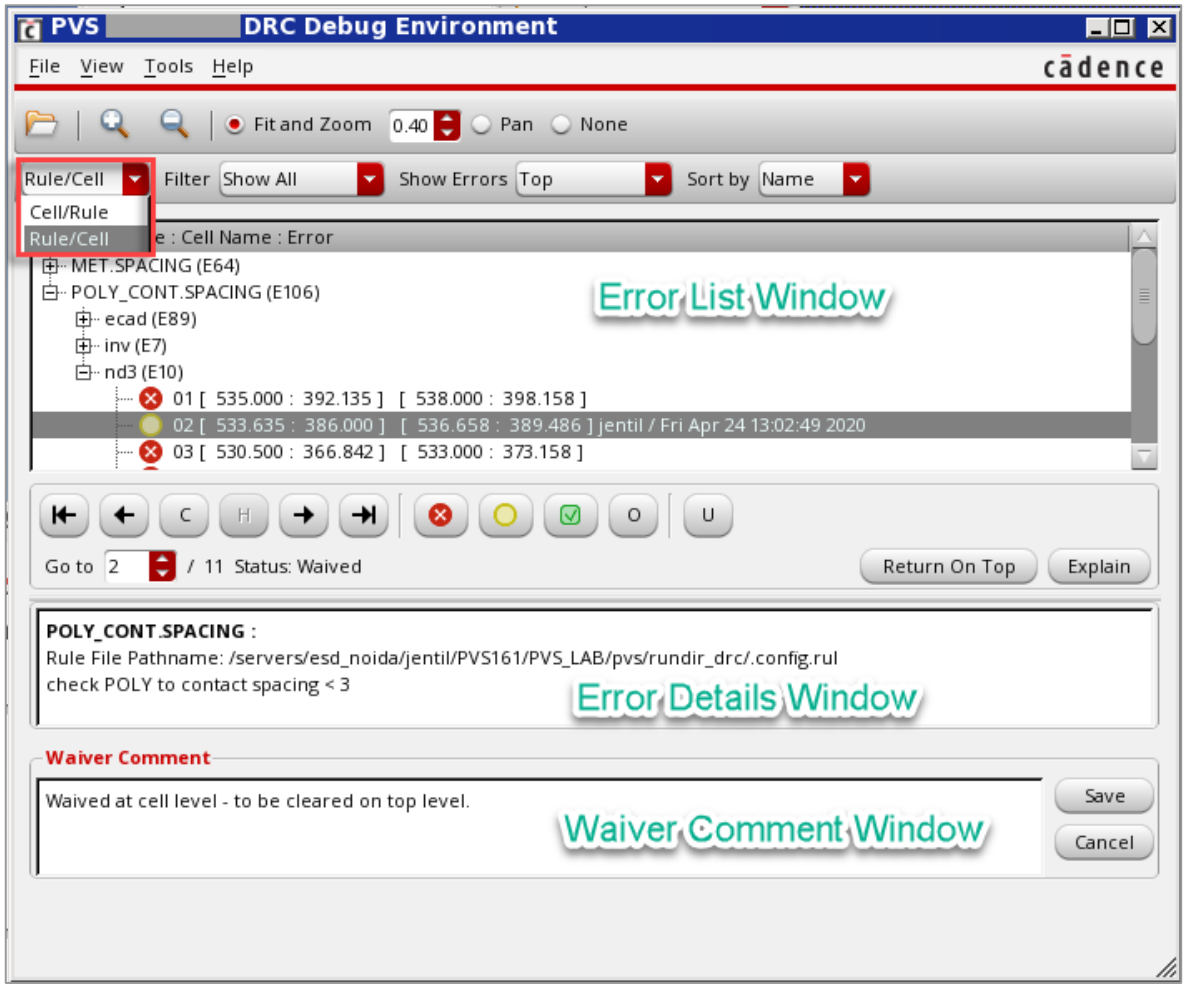


- Sets color scheme for rules/histograms, selected item in the layout viewer and the text.
- Options to automatically highlight errors in layout on or clear Previous Markers on new error marker selection.
- Set marker zoom factor, enable/disable pan/zoom options.
- To display all the errors at the cell level/top level/Edit-In-Place mode.
- Enable Property caption, show rule name and/or rule caption on error marker in layout.
- Set default tree mode (cell/rule or rule/cell), flat/hierarchical tree mode
- Exit Confirmation: Enable/disable confirmation window on tool exit.
- Compact Mode: To open DRC RV in the Compact Mode.
- Show New Result in New Tab: To open new run in a new tab.

How to Debug DRC Using DRC DE?

DRC Debug Environment is the legacy PVS DRC error browser that can be invoked at the end of the run, by selecting the DRC DE option in the Run Submission form.

1. Select list of rules – individually or by main heading – to be highlighted in the layout.
2. Chose between the Cell/Rule and Rule/Cell modes of listing.



Note

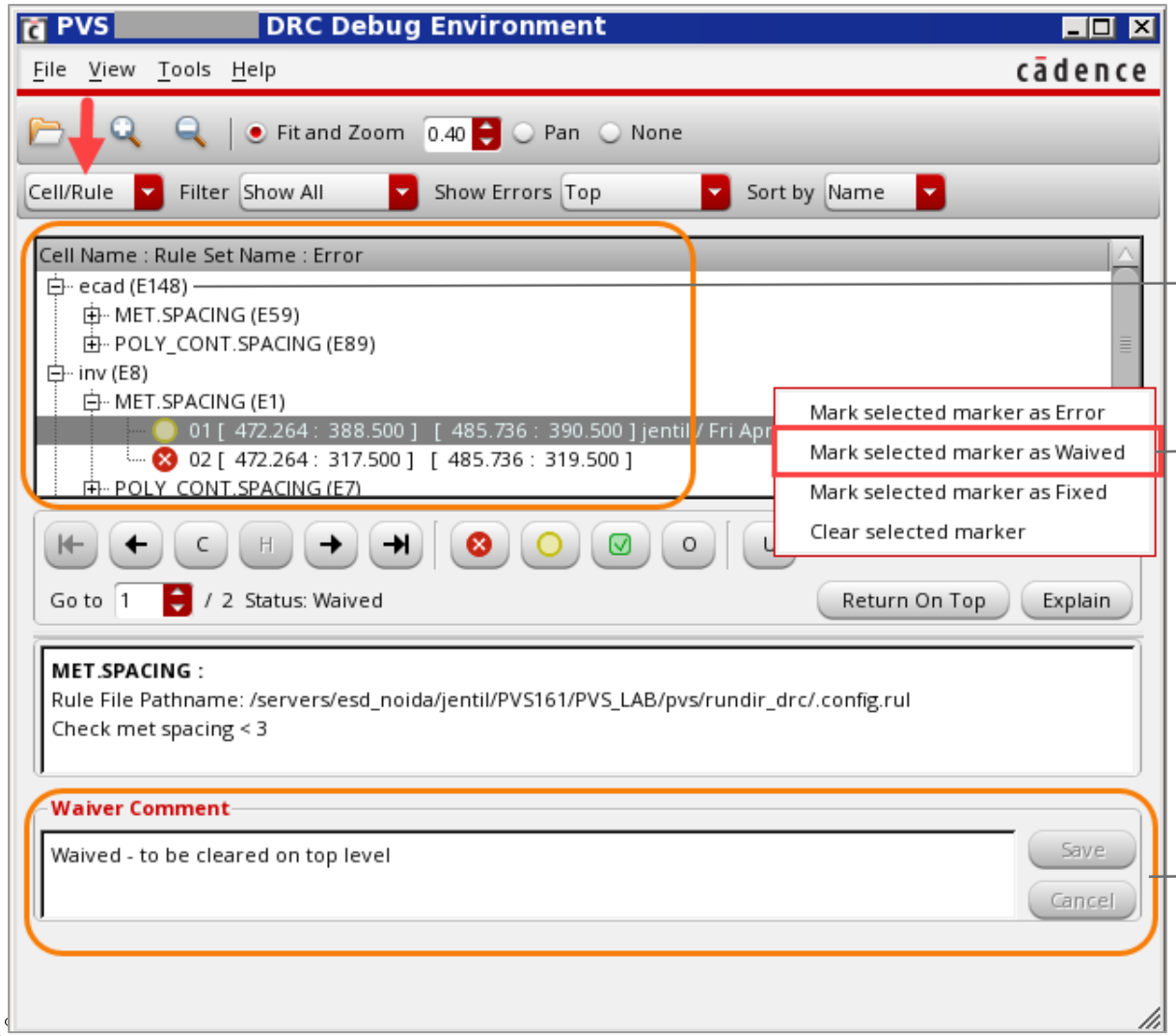
The error browser defaults to the Cell/Rule mode, where the cells are listed first, followed by the rule set names and the specific errors for each cell.

In Rule/Cell mode, the listing is based on Rule names.

Viewing DRC Errors in DRC DE

Cell/Rule Mode

DRC Run Submission Form → Start DRC DE → Submit → DRC Debug Environment



- The Cell name followed by error count – in braces – in a line.
- The user may expand the individual cell list to view the error details by clicking + sign.

- You can mark the error as error, waived, and fixed by using the right-click option.
- On the selected error, click the right mouse button. A pop-up is displayed with options

- Waiver comment window: Here add a comment to the errors marked as waived.

Viewing DRC Errors in DRC DE

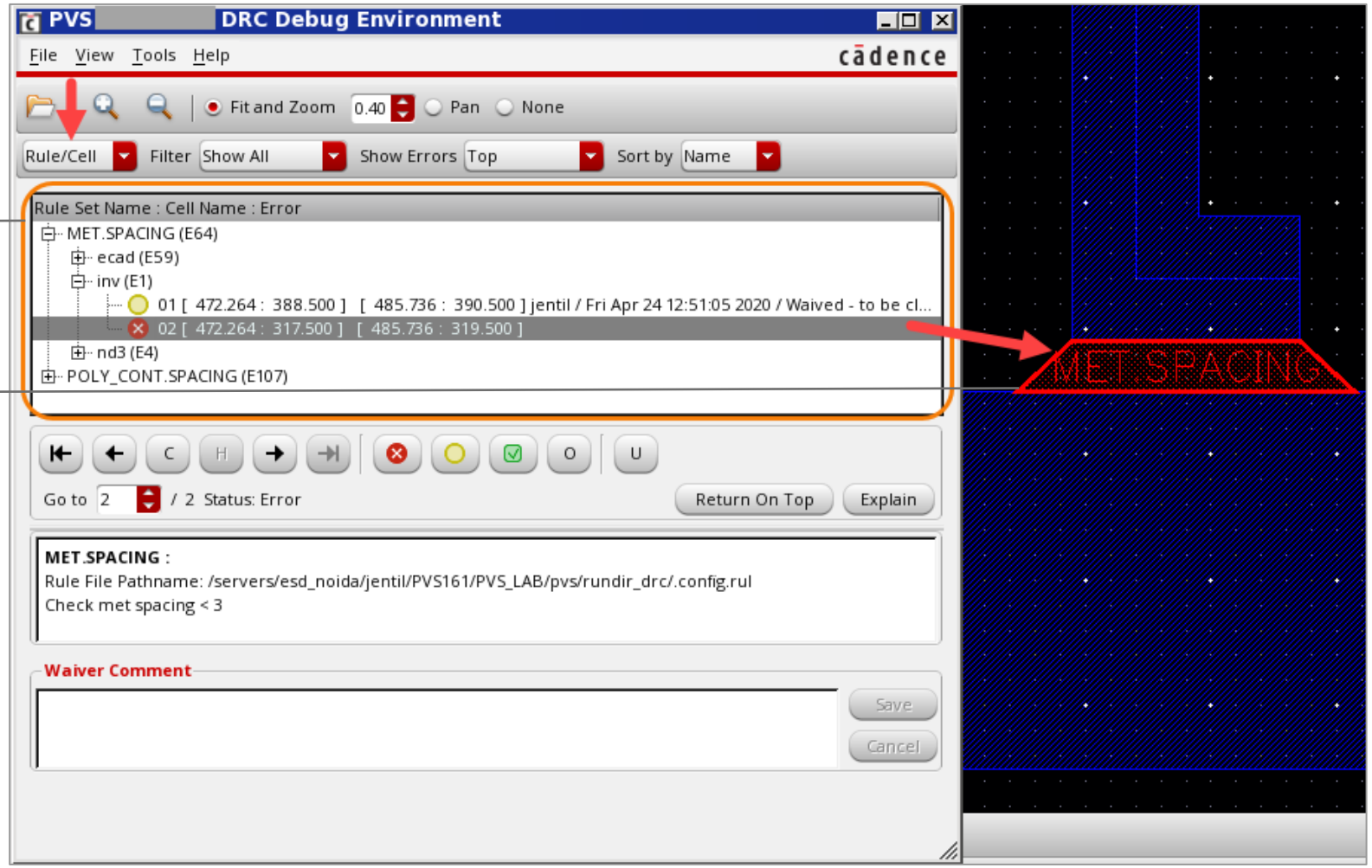
Rule/Cell Mode

In the Rule/Cell mode, DRC violations are categorized on the basis of rule names.

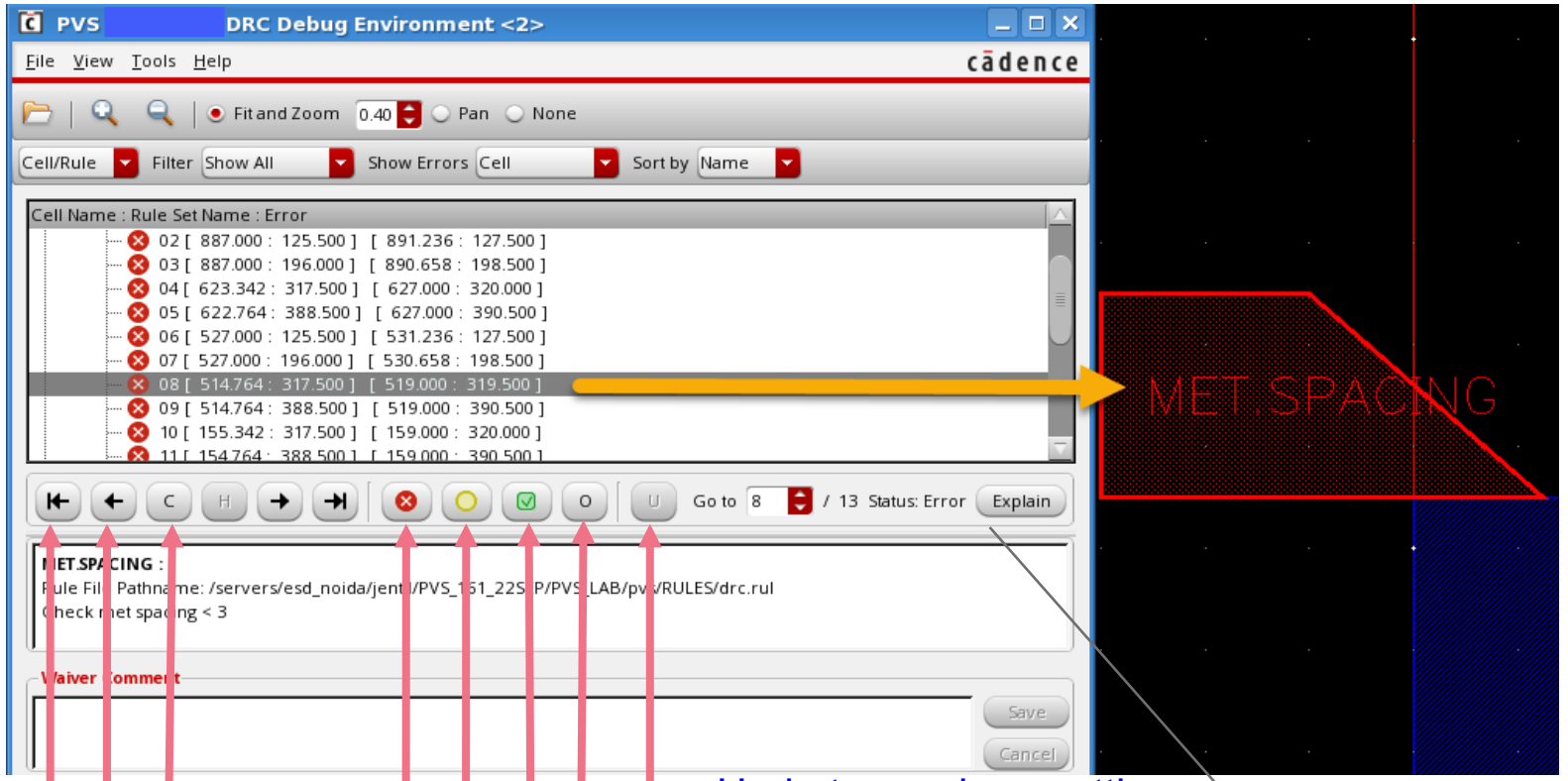
MET_SPACING (E64):
- List of cells (with cell error count).

POLY_CONT_SPACING (E107):
- List of cells (with cell error count).

- Selected error: MET_SPACING is cross-highlighted in the layout.



Debug Utilities in DRC DE



First Error

Previous Error

Clear the error marker

Error

Waived

Fixed

Owner

Undo to previous setting

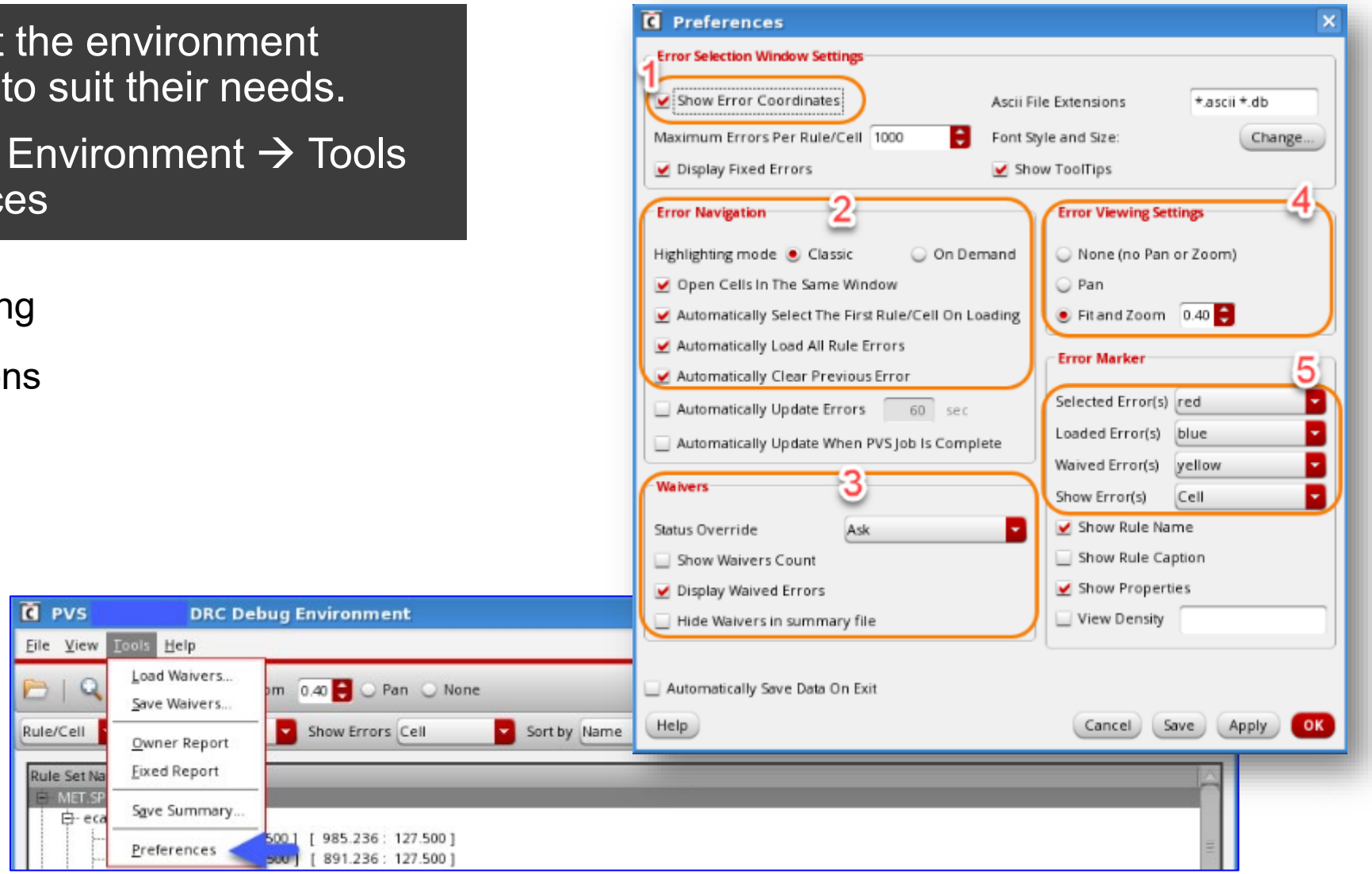
- Use the arrow keys to step through individual errors.
- Use buttons to assign Error, Waived, Fixed or Owner.
- Use the C button to clear the error.

Set Preferences for DRC DE

User can set the environment preferences to suit their needs.

DRC Debug Environment → Tools → Preferences

1. Error coordinate setting
2. Error navigation options
3. Waiver setting
4. View/Zoom options
5. Marker/Color setting

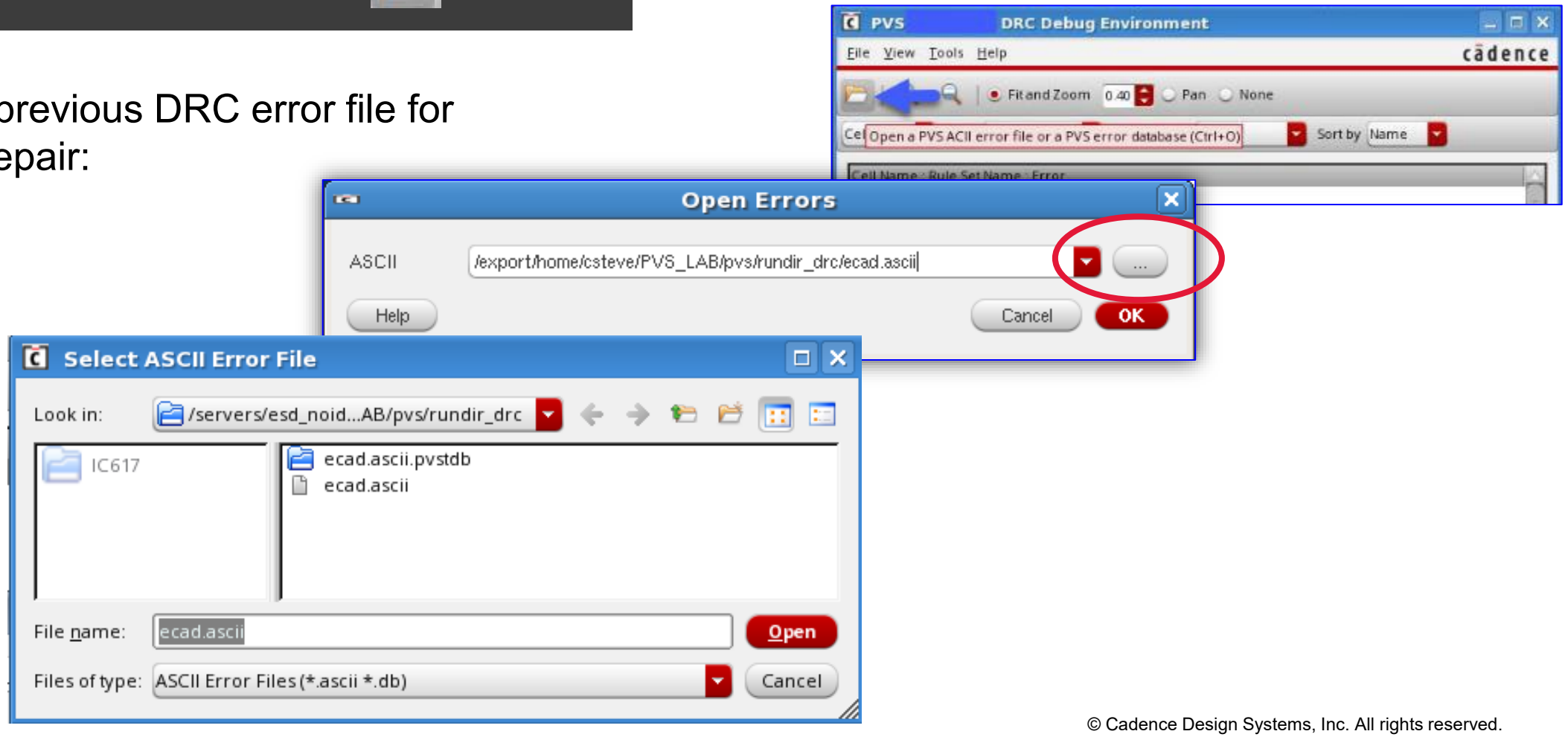


Opening Previous Error Files in DRC DE

DRC Debug Environment → File → Open (Or)

Click the Folder icon.

To load the previous DRC error file for viewing or repair:



What Is PVS Hierarchical Pattern Matching (HPM)?

PVS HPM functionality (pattern_match) searches the design for shapes in user-defined patterns (defined by pattern_shape or as flat cells in a stream (gds) file) and produces user-defined shapes at those locations where the patterns were found.

The pattern-based search engine is integrated to the Virtuoso/Innovus platforms.

- Target group of HPM: PVL rule-deck writers

Example

```
pattern_shape tee 1 0 0 0 0.16 0.11
pattern_match -filetype pattern_shape -cellname tee -bbox_sel 1 0 -overlay_sel 2 0 -orient all
```

  - Output of the `pattern_match` command is used in ruledecks as derived layer to develop DRC rules.

Advantages

- More accurate → Pattern-based DRC rule definitions are much less complicated than rule-based solutions.
- More flexible → "*Don't care*" capabilities of engine allow easier definition of target cases.
- Reduced lines of code
- Hierarchical Processing → PVS HPM engine manages all hierarchical ops → Faster & capable of processing larger DBs.

What Is the Cadence Virtuoso DFM Solution?

Cadence Virtuoso DFM solution is an in-design DFM Experience for Custom Flow implementation, which accurately assess the design's manufacturability, for both physical and electrical variability, for custom and mixed-signal designs, libraries and IPs.

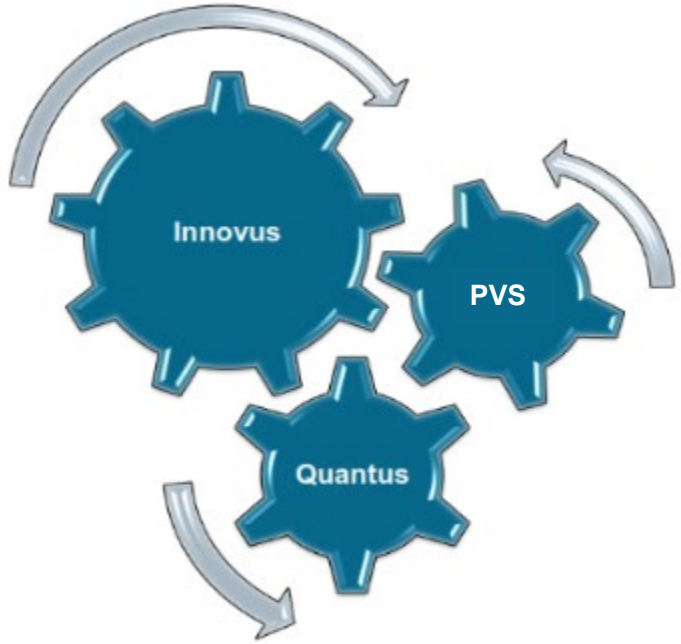
- If un-addressed, DFM leads to heavy yield loss
  - Min DRC rules fail to capture many potential yield issues.
  - Systematic shape variations result in predictable catastrophic errors:
    - Necking (opens)
    - Bridging (shorts)
- In advanced nodes (*45nm and below*):
  - LDE variability can compromise the timing, performance, and predictability.
  - DFM checks are a mandatory step in the IC design flow to address issues due to:
    - Lithography
    - Etching
    - Chemical Mechanical Polishing (CMP)
    - Mask systematic-manufacturing variations
  - DFM – No more a "nice-to-have" but a "must-have"

What Is Signoff Metal Fill Flow?

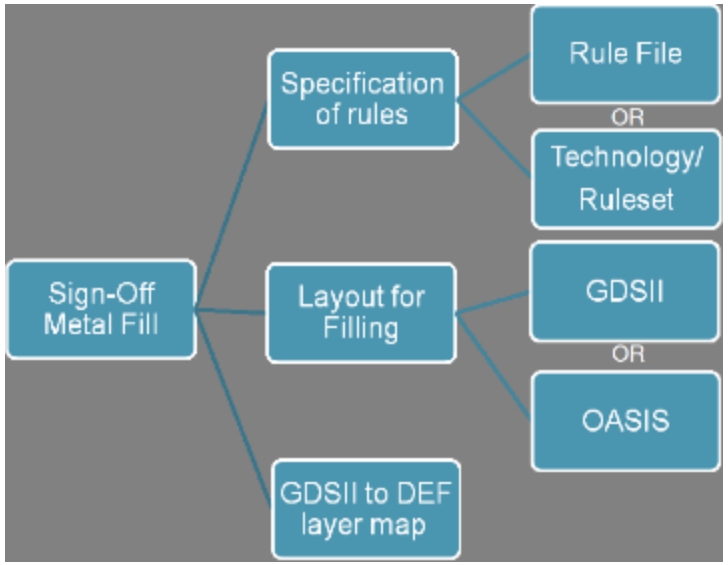
Innovus Platform

Metal Fill is the process of adding metal wire shapes (or polygons) to the design that are required to:

1. Ensure regular planarization of the wafer.
2. Achieve uniform metal density across the chip and accurate timing.



- Two Signoff Metal Fill Options in Innovus:
  1. Flat DEF Metal Fill: Default – filling only at the top level → `run_pvs_metal_fill`
  2. Hierarchical Metal Fill (HMF): Block-level filling → `add_metal_fill_signoff`
- PVS Signoff Metal Fill – minimum requirements:
  - Rules File → uses signoff rule deck to generate the metal and via fill
  - Input GDSII or OASIS file
  - GDSII-to-DEF layermap file
- PVS Signoff Metal Fill – process:
  - Runs PVS with the fill rules to create a GDSII output file
  - Converts GDSII to a DEF format file based on the GDSII-to-DEF layermap provided
  - Loads the resulting DEF file into Innovus

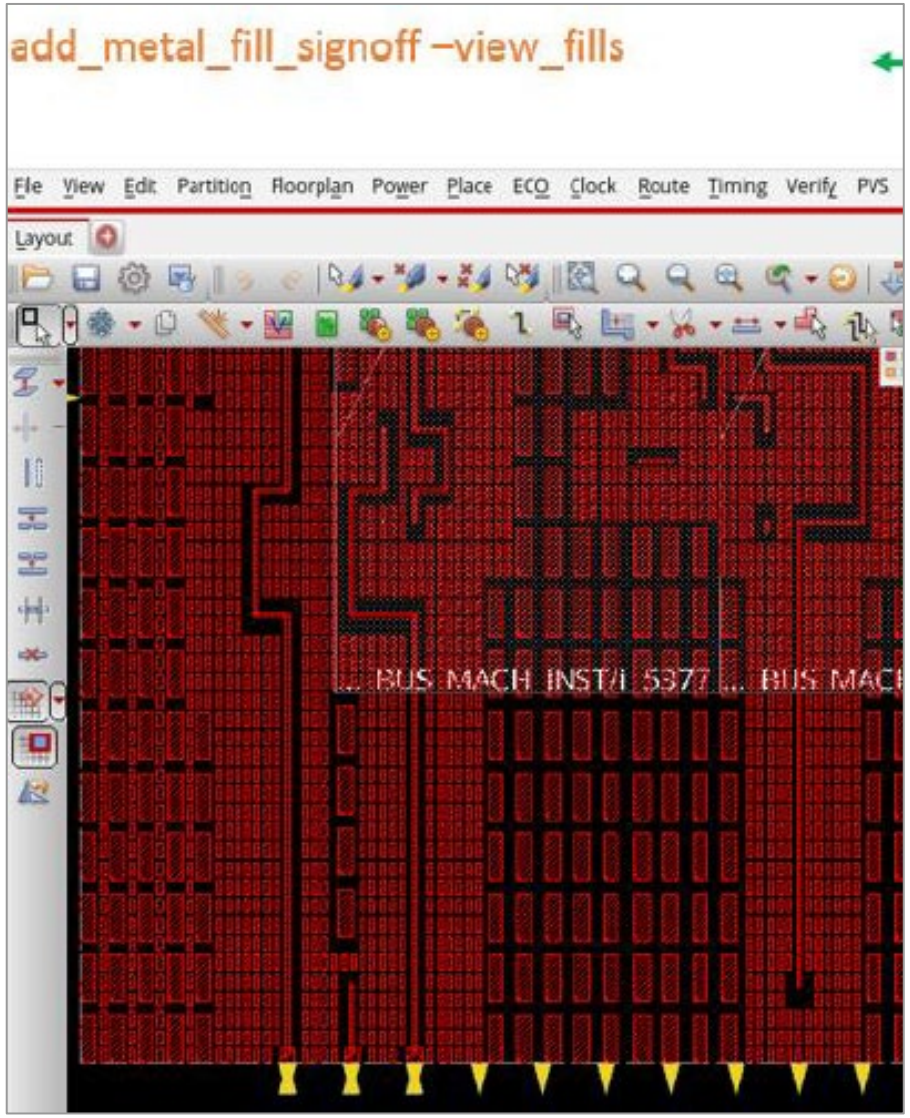


What Is Hierarchical Fill Database Flow (HMF)?

In Innovus Platform

In HMF, several fill shapes form a cell and they are instantiated to make it several levels of hierarchy. HMF DB is smaller in size due to hierarchy / multiple instantiation.

- Example: On Fullchip designs, Flat DEF metal fill final streamOut can be >100G but HMF can be <25G.
- During streamOut, HMF sits parallel to the design hierarchy. It does NOT get flattened.
- For RC extraction, HMF is provided as is (with hierarchy) to extraction tool. The extraction tool flattens it on its side.
- Why Hierarchical Metal Fill (HMF)?
  - Unified Metal Fill DB across Innovus, PVS and Quantus™
  - Greater control of individual blocks
  - Consistency throughout the design
  - Reduced DB size, memory footprint, improved runtime



Hierarchical Fill Database Flow (HMF)

HMF Commands

```
add_metal_fill_signoff {-fill | -incremental | -trim | -delete_fill | -
view_fills | -flatten | -merge_fill_edits }
```

Function	add_metal_fill_signoff option
Initial HMF	-fill: Runs signoff Metal fill on current DB view and generates HMF
Incremental HMF	-incremental: Runs Incremental Metal fill on current DB view and generates HMF
Trim Fill – Net & Slack	-trim: Trims Metal fill for timing on current DB view and generates HMF
Delete HMF	-delete_fill: Deletes fills on select layers in a given window or coordinates

Function	add_metal_fill_signoff option
View/Edit HMF	-view_fills: View fills in Innovus (cannot be edited) -flatten: Flattens and loads fill shapes into Innovus -merge_fill_edits: Merges user edits to HMF by user -delete_flatten_fills: Deletes flattened fills without updating HMF database.
HMF in BG /LSF modes	–bg –master_lsf –lsf

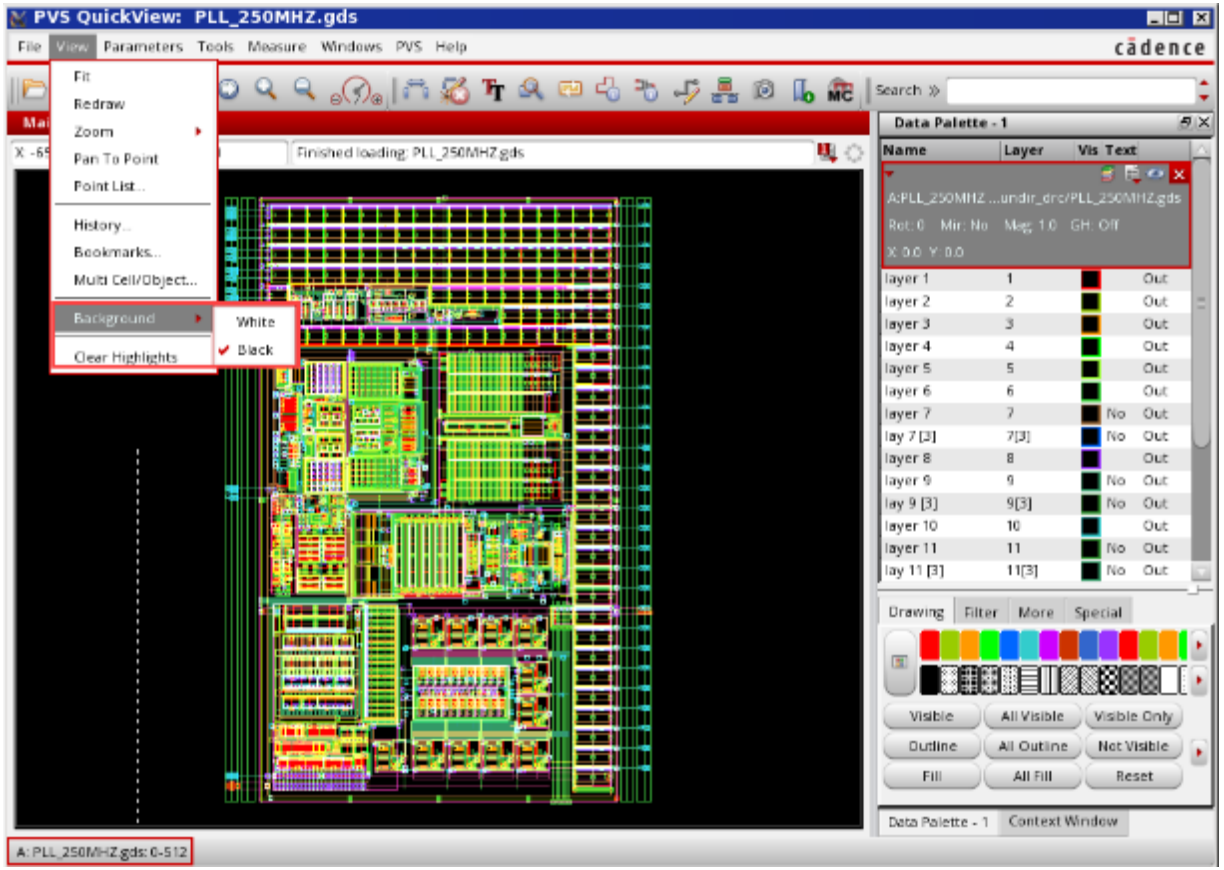
Example

```
add_metal_fill_signoff -fill -bg -master_lsf -lsf -temp_working_dir LSF_RUN1
```

What Is PVS QuickView?

PVS QuickView is Cadence Design Systems standalone data and results viewer. PVS QuickView offers design and manufacturing teams a standalone, faster, easy-to-use, high-performance and extensible environment for efficient tapeout and chip finishing.

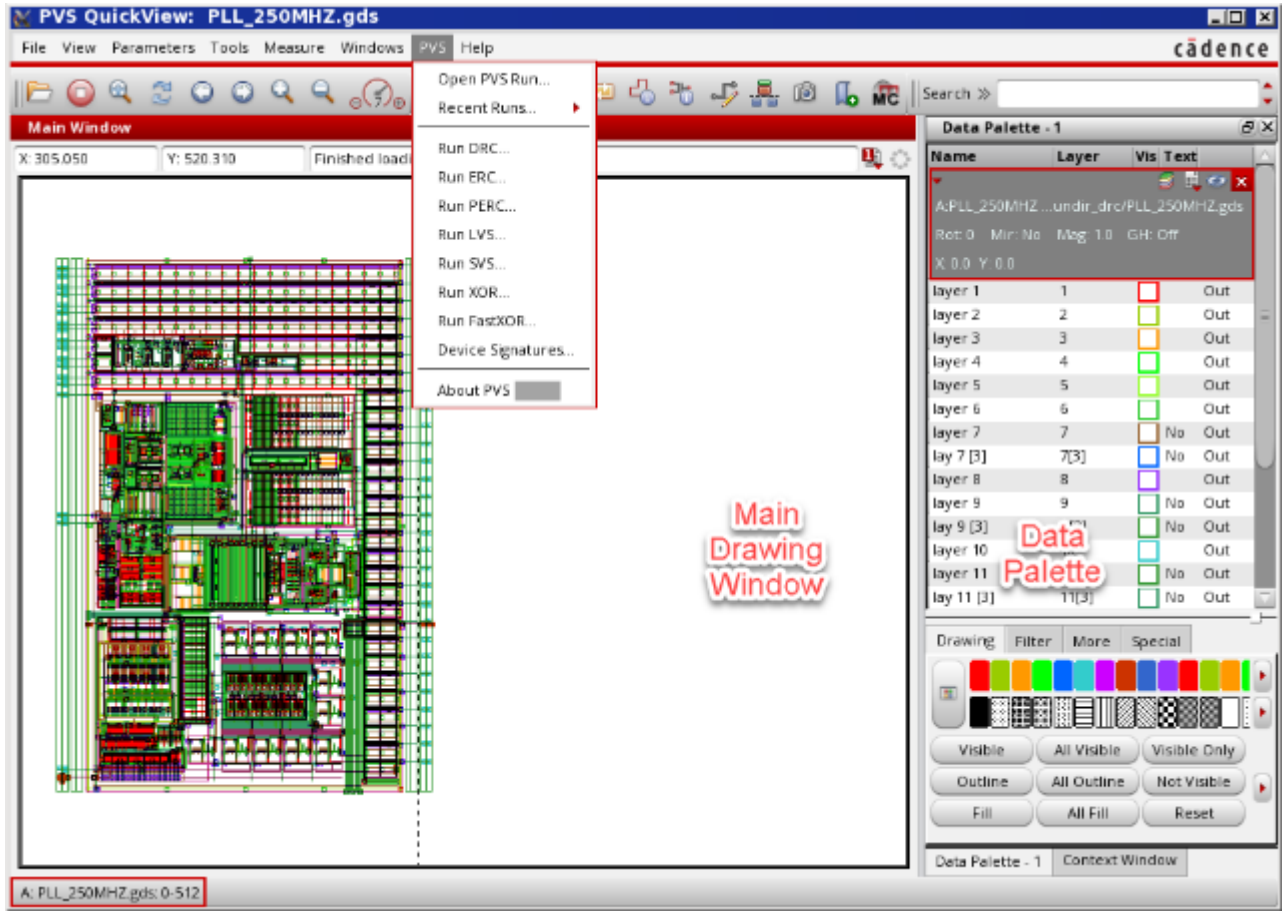
- Tightly integrated with the Cadence PVS platform.
- Loads large layouts in seconds.
- Central job submission, high-capacity/multi-format viewing, analysis, debugging and signoff cockpit for PVS flows – DRC, XOR, LVS, ERC.
- Debugging features of QuickView:
  - Net connectivity tracing, visualization, overlay, and GDSII/OASIS editing.
  - Draw, examine, compare and validate IC data in various layout and pattern data formats.
  - Interactive error browsing and cross-probing.
- Imports and applies Virtuoso Tech files to loaded data.
- Point-to-point path analysis through identified connected geometries.



How to Start PVS QuickView?

```
K2_VIEWER_HOME/bin/k2_viewer
```

- All PVS forms and access to PVS rule decks available as in Virtuoso environment.
- Environment Variable: K2_VIEWER_HOME must be set before you can invoke Quickview.
- Two main parts of the GUI:
  - Main drawing window
  - Data palette



A series of videos on various *QuickView* features are available at the COS2.0 PVS QuickView channel.

Frequently Asked Questions
PVS DRC

Can command-line DRC results be loaded in GUI DRCDE?

Always run PVS with **-ui_data** to load the results in GUI.

```
pvs -drc -ui_data -gds gdsFile -top_cell topcellname <main rule file>
```

How to ignore a cell/block during the DRC check?

Use the **exlucde_cell** command. Example is shown in the box.

```
exclude_cell NAND*
```

How to select / unselect a rule in PVS DRC check?

1. Create a text file with rules to be selected/unselected (say **select.pvl**); wildcards supported:

```
select_check -drc M1*
select_check -drc CPO* PO*
unselect_check -drc M1* M2*
```

2. Provide **select.pvl** as a last argument for DRC run:

```
pvs -drc -gds gdsFile -top_cell topcellname <main rule file> select.pvl
```

Layout Versus Schematic (LVS)

What Is Layout vs. Schematic (LVS)?

LVS process guarantee that the layout represents the circuit you desire to fabricate.

- CDL is extended version of SPICE – both are mutually compatible.
- PVS supports mixed netlists – in SPICE+CDL+Verilog formats and are order independent.
- But if you provide Verilog netlist; v2cdl convert Verilog to CDL netlist and then PVS uses it.

- The LVS process.
  1. Extraction: Recognize drawn device shapes and connectivity info from layout; then extracts a netlist out of it.
  2. Comparison: Compare the layout and schematic netlists to identify any discrepancies.
- Custom log file creation:
  - Use the following command, to direct LVS to dump, runtime information to a log file, for future reference:
    - `pvs -lvs rule_file |& tee log_file`

LVS Flow – Extraction, Reduction and Comparison

- Extraction:
  - To extract drawn layout layers, devices, nets, connectivity info from the layout geometry data input to LVS.
  - Runs layout DB with polygon areas through area based logical operations to define device recognition layers, device terminals, wiring conductors and vias and pin locations to determine the components represented in the layout.
  - The layers (metals or vias) that form devices can have various measurements performed to and these measurements can be attached to these devices.
  - Extraction report (*reportName*) summarizes extraction process including reports on issues about stamping conflicts, diff labels on same net (a common issue), malformed devices etc.
- Reduction:
  - An intermediate step between Extraction and comparison.
  - The tool combines the extracted components into possible series and parallel combinations and generates a netlist representation of the layout (extracted netlist e.g., SPICE).
  - A similar reduction is performed on the "source" Schematic netlist.
- Comparison:
  - Compare the extracted layout netlist to the schematic netlist and reports differences.
  - If the two netlists match, then the circuit passes the LVS check.
  - Mathematically, the netlists are compared by performing a Graph isomorphism check to see if they are equivalent.
  - LVS Comparison report (*reportName.cls*) summarizes the differences and other info about compared cells.

Run PVS LVS from the Command Line

Batch Mode

```
pvs -lvs [lvs options] rule_file1 [ ... rule_fileN]
```

LVS Options and Arguments: *Examples*

Option	Definition
`-gds gdsFile`	Specifies the layout database for the PVS run
`-control* <file_name>`	Specifies the control file created by PVS GUI
`-h | -help`	Displays the available command-line options
`-layout_top_cell cellName`	Specifies the layout top cell name for the PVS run
`-source_top_cell cellName`	Specifies the source top cell name for the PVS run
`-rc_data **`	Specifies to generate output data for the Quantus™ tool
`-ui_data | -no_ui_data`	Specifies to generate (or not to) the output data for the error browser
`-dp n`	Runs in local mode, n: number of processors
`rule_file1 [.. rule_fileN]`	File(s) which contain the design rule checks

Tips

- In the batch mode LVS run, as the files generated are placed in the current working directory (`cwd`), it is better to make separate run directories and invoke the runs from within each run directory, to keep an order.
- Be sure any source files listed in your rule file are linked to the right paths.

Running PVS LVS from the Command Line

Batch Mode (continued)

Here are a few examples of running LVS from the command line.

```
pvs -lvs [lvs options] rule_file1 [ ... rule_fileN]

pvs -lvs -ui_data -rc_data -control lvsCtrl 65nmlvs.rul

pvs -lvs -rc_data 65nmlvs.rul optional_pvl.rul

pvs -lvs -gds test.gds -top_cell test -source_Verilog top.v -source_cdl stdcells.cdl
-source_top_cell test_sch 65nmlvs.rul optional_pvs.rul
```

* *-control <file_name>*

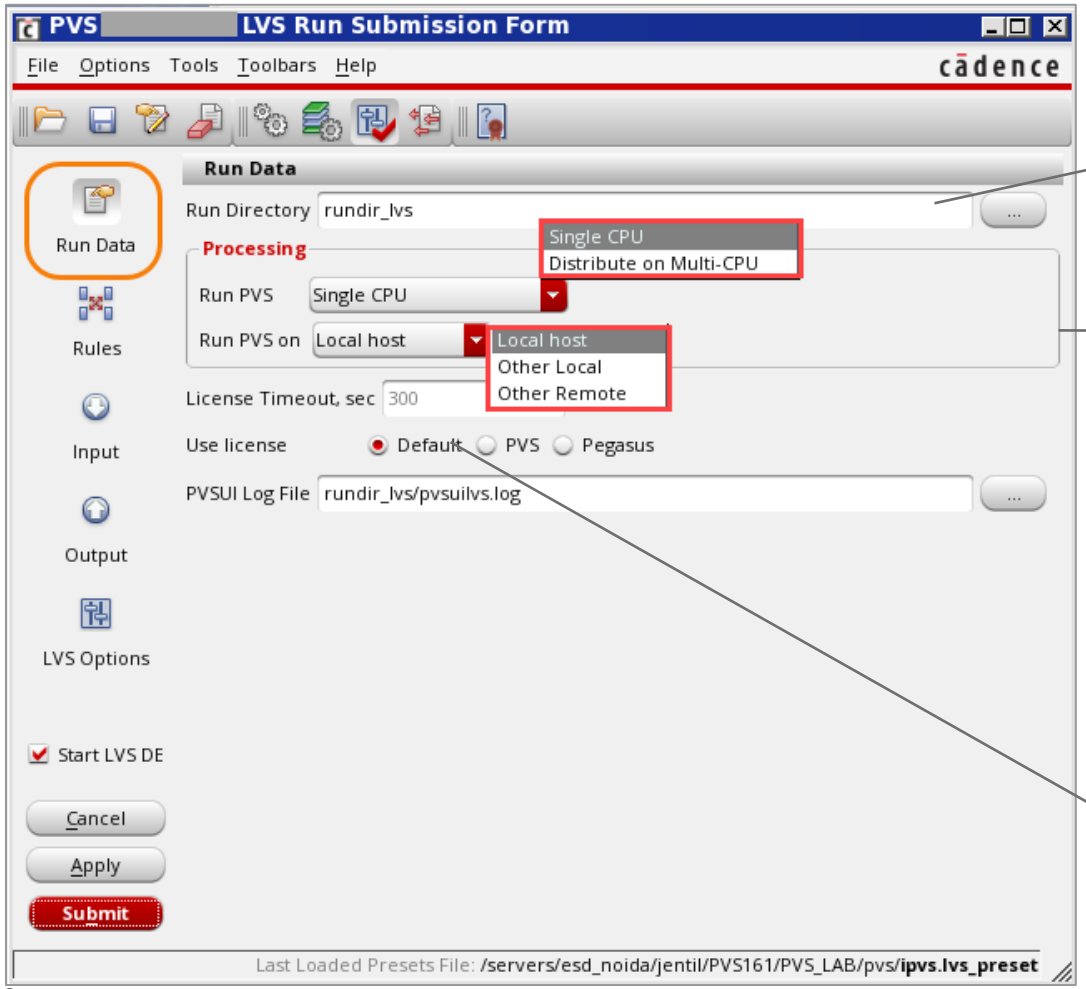
- This option can be used if you have a run directory for the run, which was submitted from the GUI.
- Control file, created automatically when a job is submitted from GUI, can be used to rerun PVS job from the command line with exact same options as in the GUI submission.
- It is placed in the run directory and named `pvslvsctl`.

-rc_data/-no_rc_data

- This argument is used to generate (or not to generate) output data for the Quantus extraction tool.
- If the `lvsrule` deck contains the `mask_svdb_dir <dir_name>` command, PVS creates the data for Quantus in that directory, otherwise it creates in the LVS run directory.

Setting Up CPU Processing in PVS GUI

LVS Run Submission Form → Run Data → Processing



PVS combines single processor performance with distributed processing/multithreading to accelerate DRC/LVS turnaround of advanced designs.

- Specify Run Directory.

Set up Processing: Select number of CPUs & machine.

- Run PVS: Select PVS run mode
  - Single CPU (default) & Distribute on Multi-CPU (DP) modes.
- Run PVS on options:
  - Local host: Run on the local machine (default).
  - Other Local: Run as a part of a batch script on local machine.
  - Other Remote: Run from a Remote machine.
  - SSH/RSH (in DP mode only): Secure/Remote shell modes.

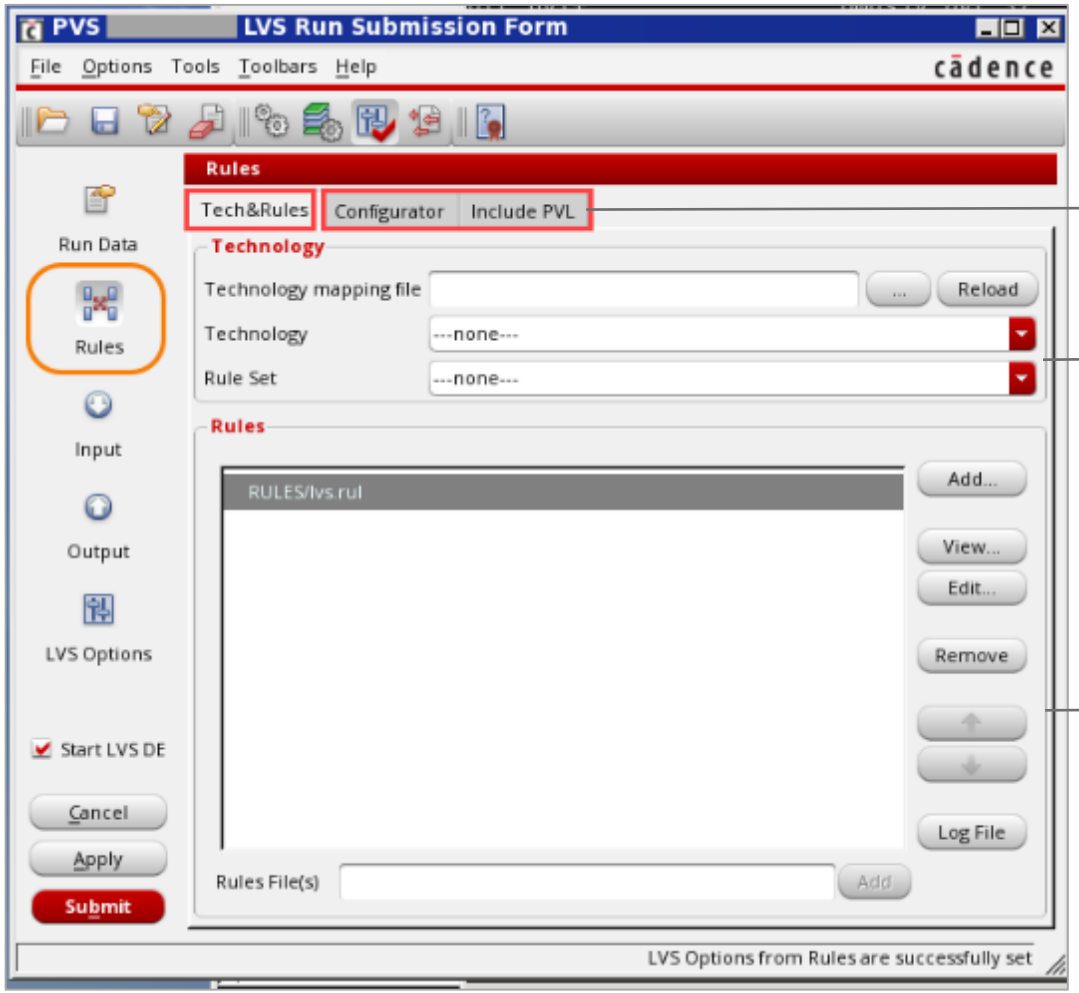
- License Timeout, sec – PVS has license queuing. If all licenses are in use, then PVS waits for them to free up.
- Default is 300 seconds.

LVS Run Submission Form

Rules

Select the appropriate technology and rules for the PVS LVS run.

PVS LVS Run Submission Form → Rules → Tech&Rules / Configurator / Include PVL



Configurator
- To customize the PVS GUI.

Include PVL Rules
- Supplement or Override existing rules.
- Enable the Include PVL Rules entry field.

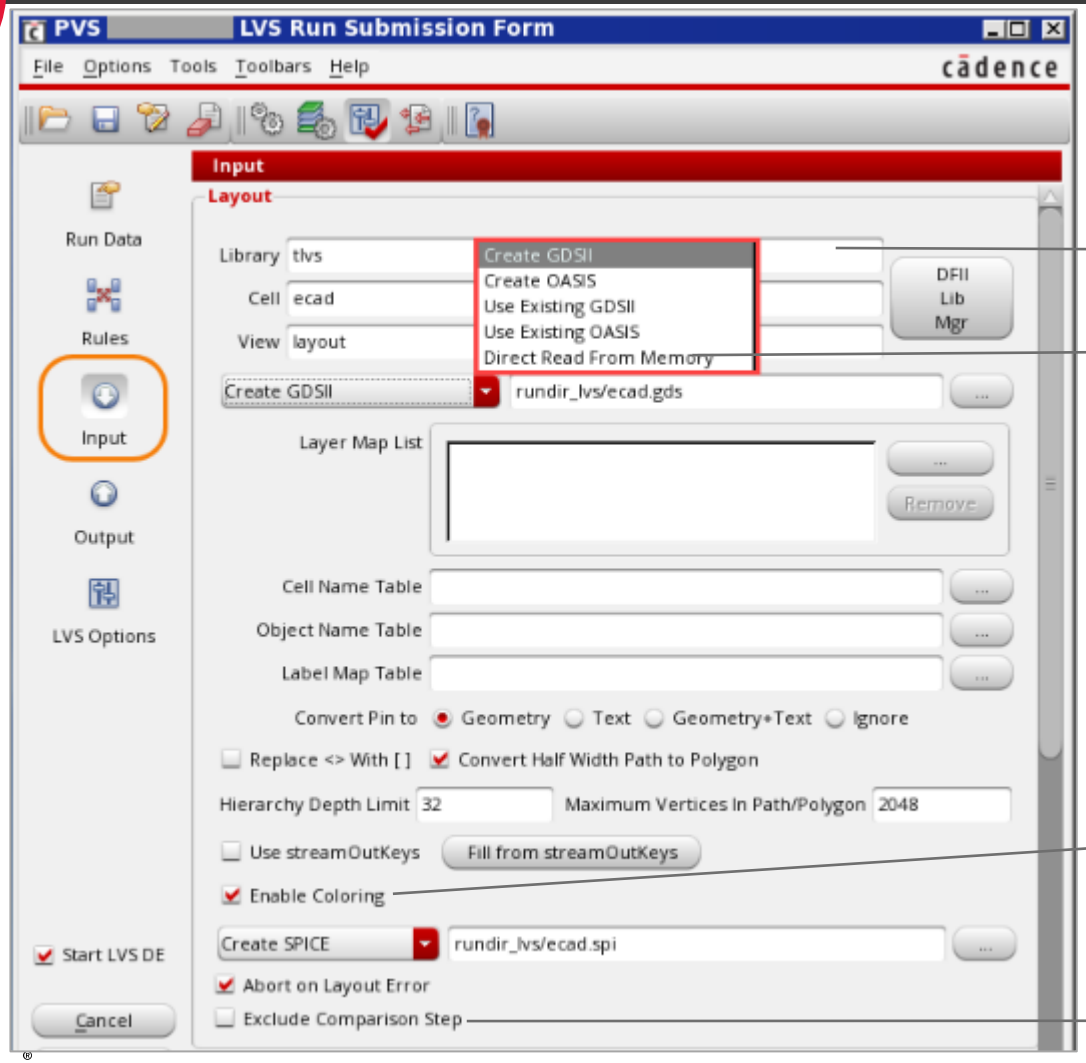
Technology
- To set the relevant Technology mapping file/Technology and rule set.
- Applicable for foundry technologies.

Rules
- To set custom rule deck files.

LVS Run Submission Form

Input Layout

PVS LVS Run Submission Form → Input → Layout



Layout: Cellview information
- Library, Cell and View
- Select DFII Lib Mgr to browse to layout cellview.

Layout Input Format options:
- Create GDSII file
- Create OASIS
- Use Existing GDSII
- Use Existing OASIS
- Direct Read From Memory

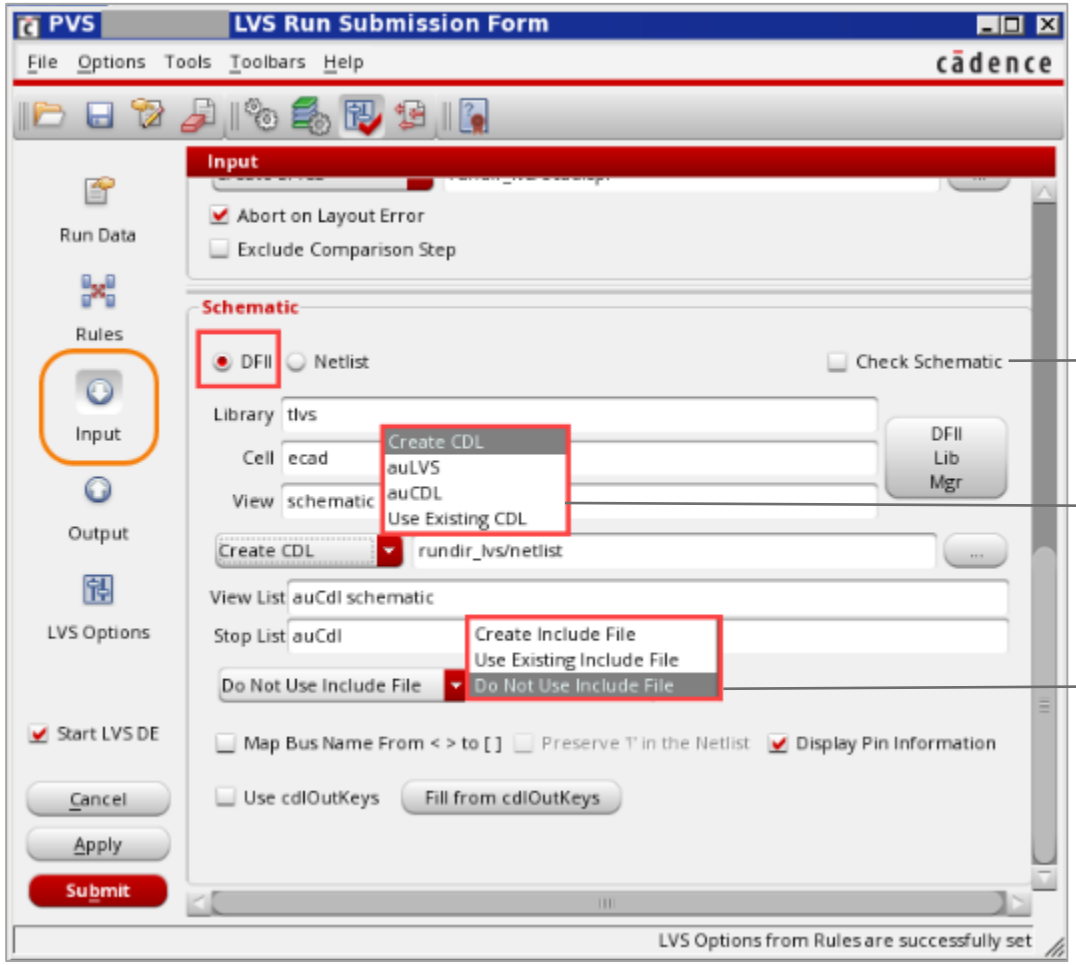
- Enable Coloring – Only for Adv nodes (ICADV 12.3+ ISR releases).
- Virtuoso® passes the predefined mask data to PVS.

Exclude Comparison Step: (pvs -lvs -no_compare ..)
- Launches the ERC instead of LVS run.
- Disables all options related to comparison step.

LVS Run Submission Form

Input Schematic – DFII

PVS LVS Run Submission Form → Input → Schematic (DFII)



Check Schematic: Read schematic netlist to check its integrity.

CDL netlist creation options:

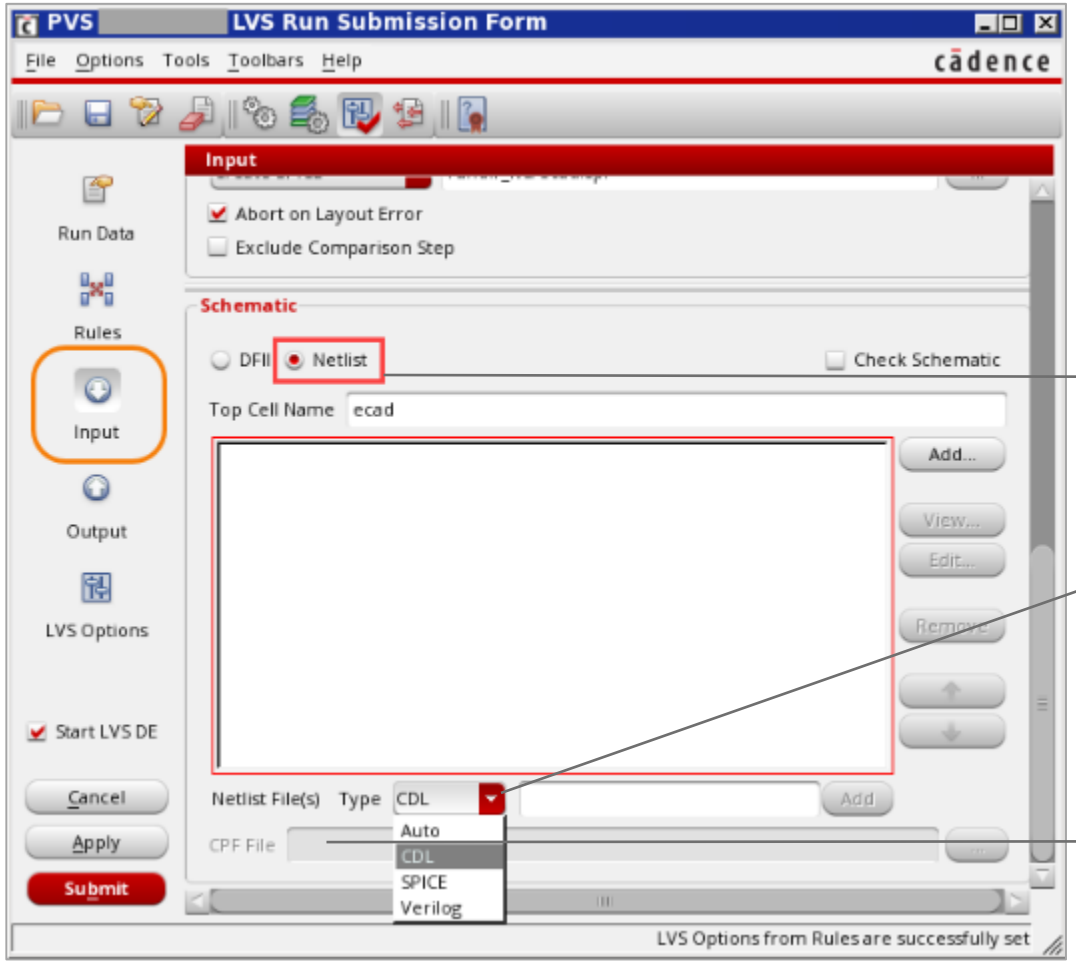
- Create CDL
- auLVS (PVS internal netlister)
- auCDL (PVS internal netlister)
- Use Existing CDL

- To include additional Verilog, CDL or SPICE file to be used along with DFII database.

LVS Run Submission Form

Input Schematic – Netlist

PVS LVS Run Submission Form → Input → Schematic (Netlist)



Select Netlist

Netlist type options:

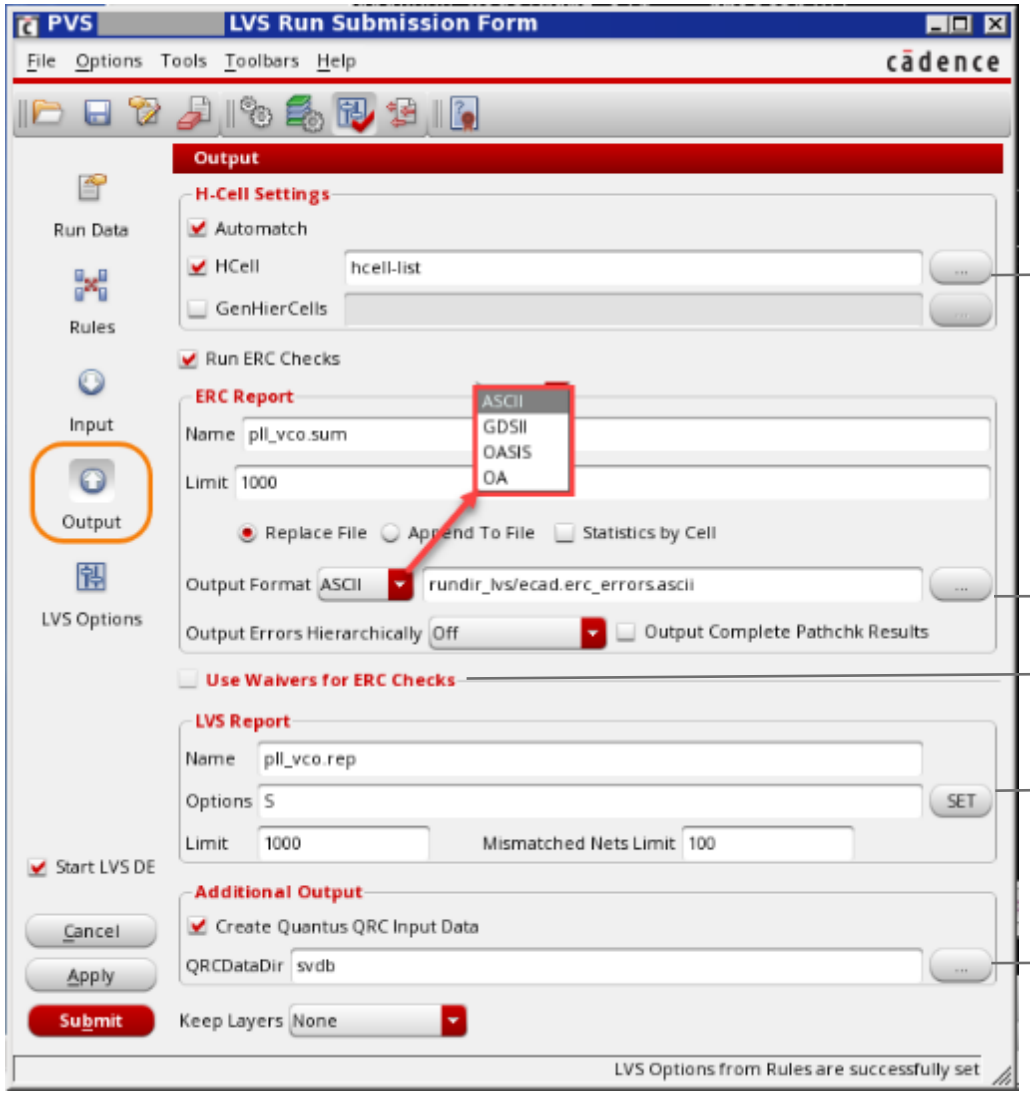
- Auto
- CDL
- SPICE
- Verilog

- CPF (Common Power Format) file to be used during the LVS run. Active only if for Verilog netlist.

LVS Run Submission Form

Output

PVS LVS Run Submission Form → Output



- **H-Cell Automatch**: PVS tries for automatic cell name bindings (PVL command: automatch).
- If Automatch not used, cells outside hcell file are flattened.
- Reports violations at top level – difficult to debug.

- **H-Cell**: PVS use hcell file for cell name binding.
- Recommendation: Use Hcell file – so that LVS violations gets reported at cell level.

- **GenHierCells**: To have PVS generate (with user script) own hcell list for cell name binding.

Options ERC run: Report Name, output Format, limit etc.

Waivers for ERC checks, if any.

LVS Report:

- Name of LVS Extraction report.
- SET the Output options.

- For PVS to Create Quantus Output Data.

PVS Bindings by Name During LVS

Automatch

PVS LVS Run Submission Form → Output → H-Cell Settings → Automatch

- When LVS is run with the **`Automatch`** setting in the GUI:
  - The tool automatically binds cells and pin names in the schematic to identically named cells in the layout.
- If this matching is incorrect, you can use an ASCII text file (**`Hcell`** option) to:
  - Create bindings that are not based on name.
  - Break bindings between identically named cells.

  An example is given on RHS.
- How to create bindings between nets in layout and schematic:
  - **`lvs_cpoint`**: This PVL command creates bindings between nets in layout and those in schematic for LVS comparison.
  - In a Verilog netlist, data bits are enclosed by square brackets.
  - If the data bits in the layout netlist are enclosed by angle brackets, a wildcard binding such as the one shown right can map the data bits in the layout netlist to the data bits in the source netlist.

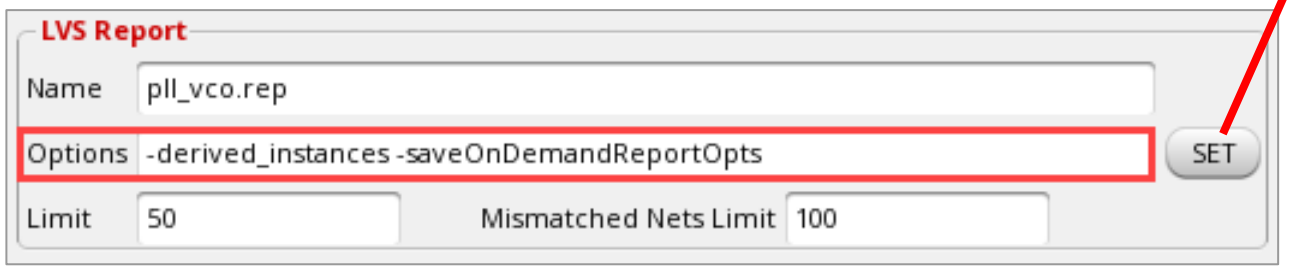

```
hcell layoutCellName sourceCellName
hcell inverter INV  ;; maps layout cell 'inverter' to
schematic cell 'INV' schematic cell
```

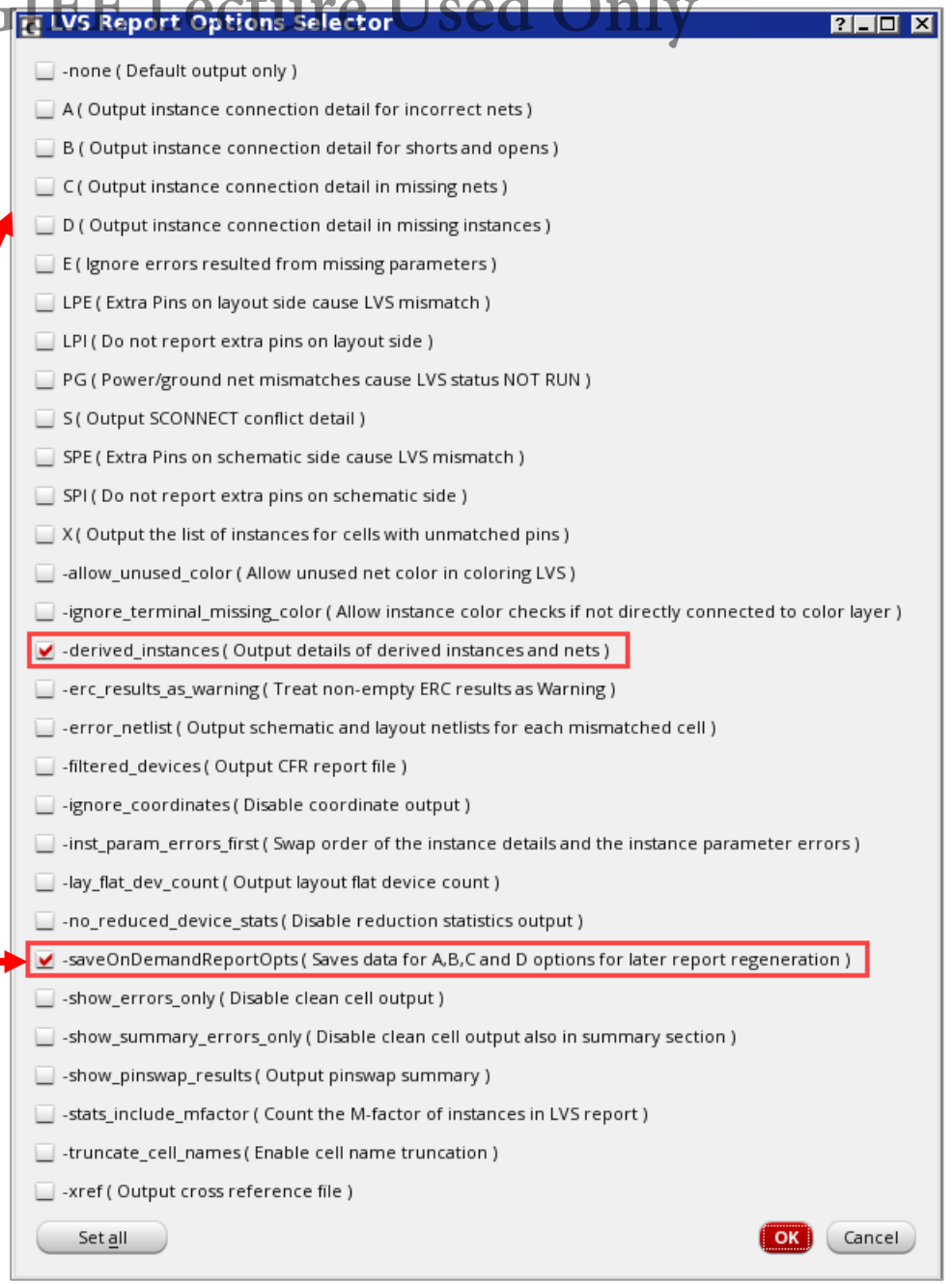
```
lvs_cpoint layNetName schNetName
lvs_cpoint vss "gnd!"          ;;this binds 'vss' in
layout to 'gnd!' in schematic
```

```
lvs_cpoint "*<*>" "*[*]"
```

PVS LVS Output Report Options

PVS LVS Run Submission Form → Output → LVS report → SET → LVS Report Options Selector





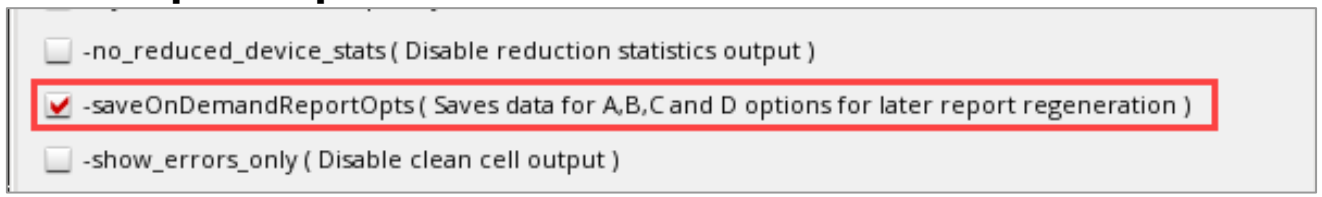
To regenerate LVS reports with options A,B,C, D, enable this checkbox:

`-saveOnDemandReportOpts`

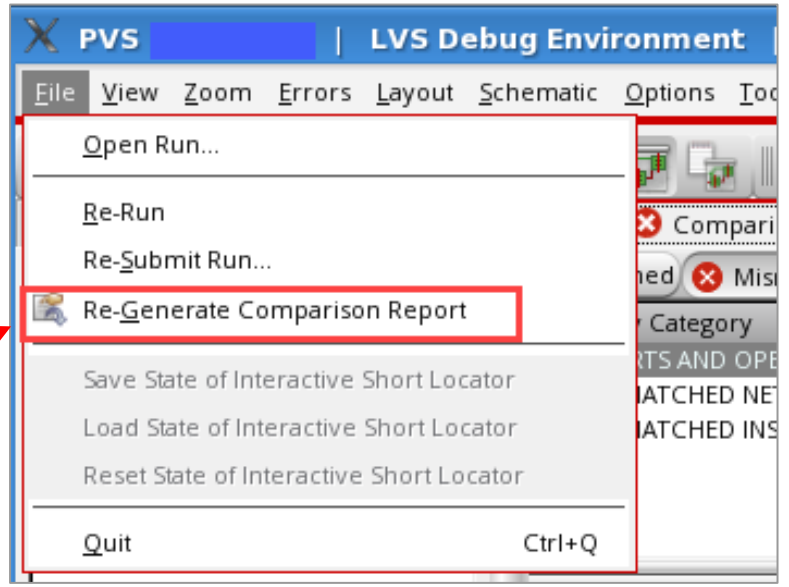
PVS LVS Output Report Options

Example: saveOnDemandReportOpts

LVS Report Options Selector Form

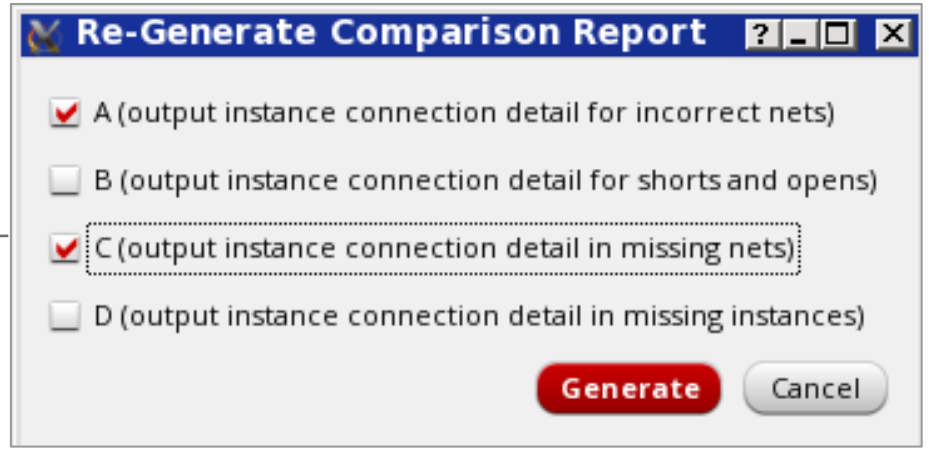


PVS LVS DE → File → Re-Generate Comparison Report



The Re-Generate Comparison Report feature is activated ONLY if LVS is run with option **`saveOnDemandReportOpts`**.

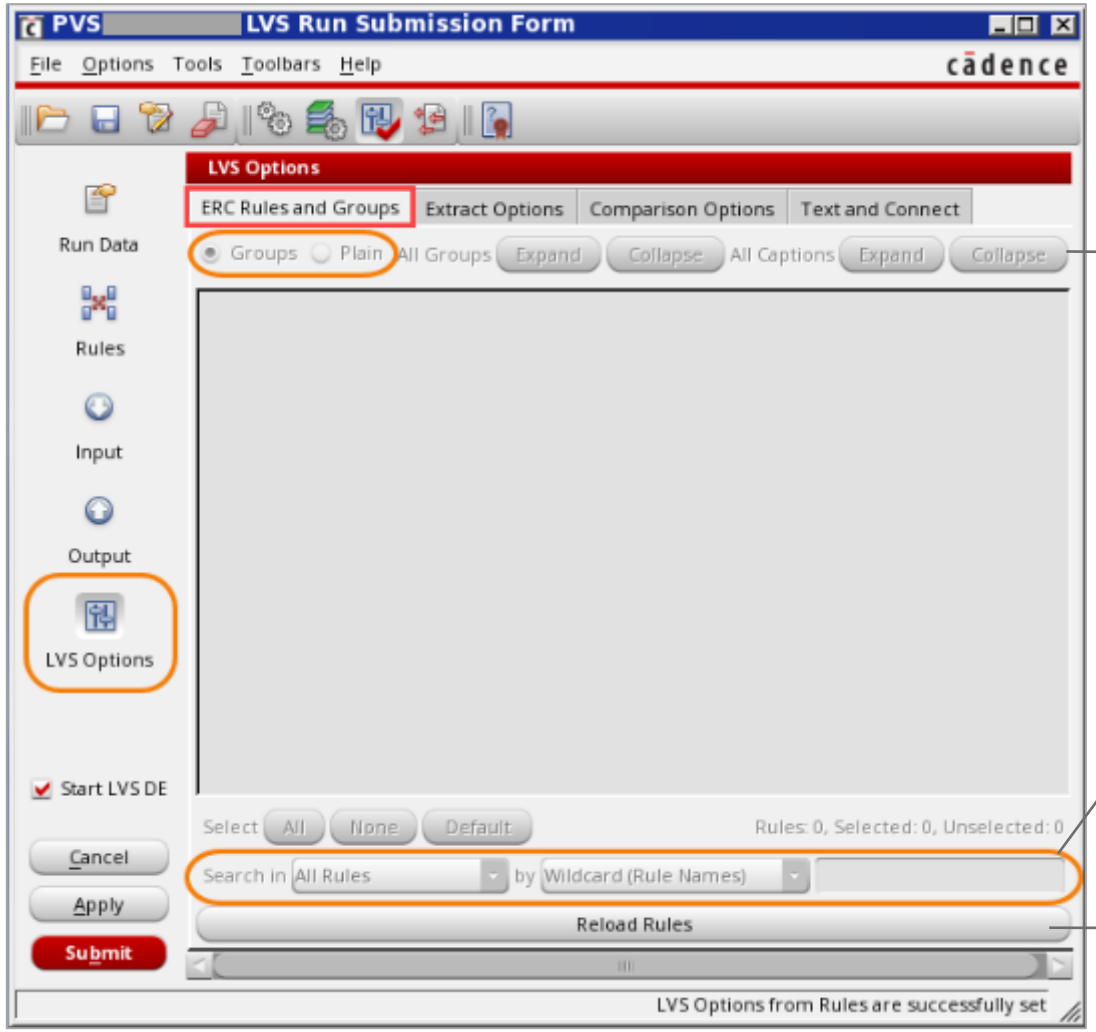
- Options A,B,C, D check-boxes are available for next round of report regeneration.



LVS Run Submission Form

LVS Options – ERC Rules and Groups

PVS LVS Run Submission Form → LVS Options → ERC Rules and Groups



- List rules in group/plain.
- Expand/collapse the list of rules.

Customize your search results using various drop-down options:

- All Rules, Connectivity Rules and Density Rules.
- Wildcard options: (Rule Names, Rule Captions).
- Reg. Exp. options: (Rule Names, Rule Captions).

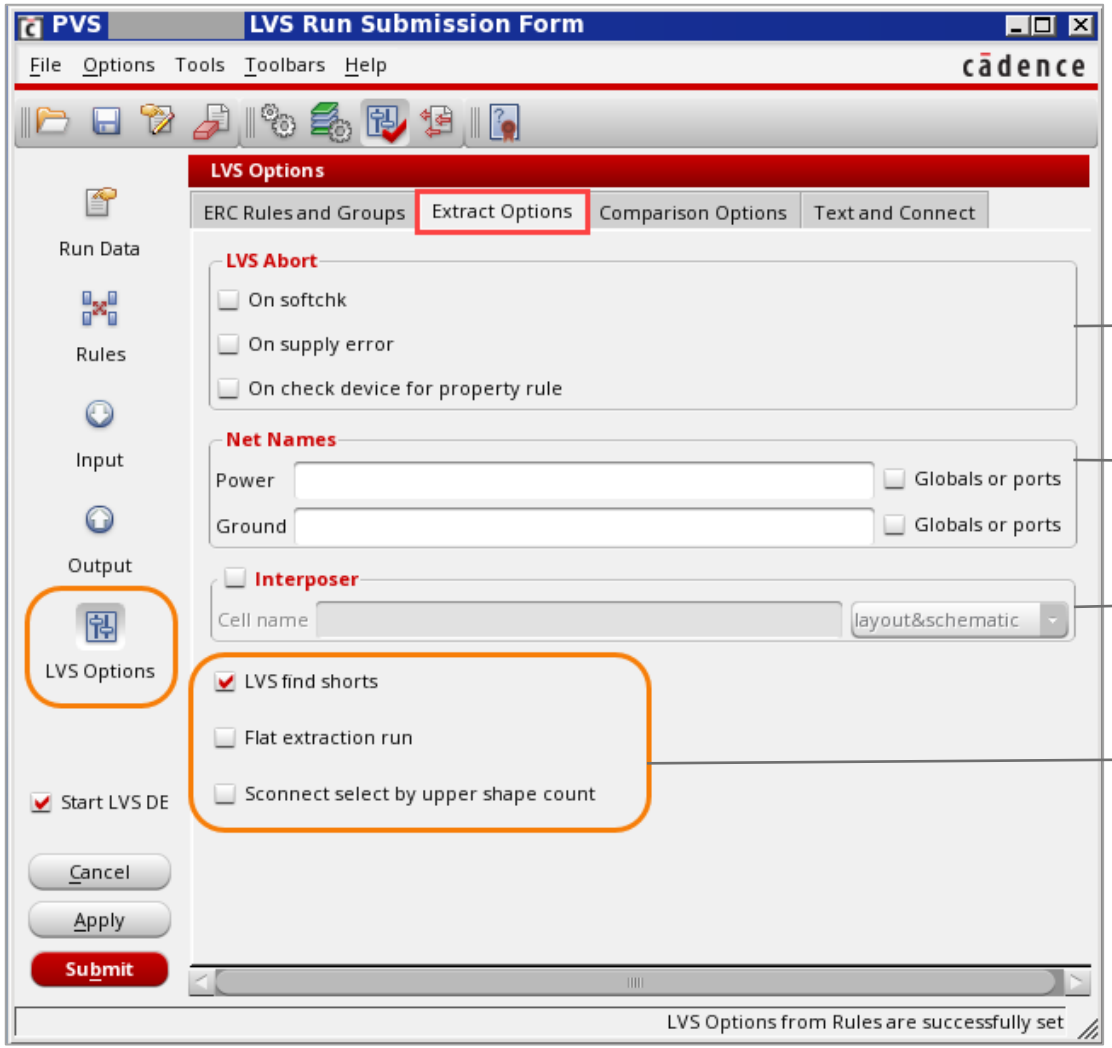
Reload Rules

- Changes made to the rules previously loaded will be refreshed by clicking the **Reload Rules** button.

LVS Run Submission Form

LVS Options – Extract Options

PVS LVS Run Submission Form → LVS Options → Extract Options



LVS Abort On:

- Softcheck (lvs_softchk): Abort on Soft Connect errors
  - Soft Connect: A node connected through non-routing layer (Diff, N-Well) – Highly resistive & poor circuit performance.
- On supply (PG) error or device property rule error.
- In *lvs_abort* cases, LVSDE Run Status form reports comparison status Abort with a red icon .

- Specify the PG net name(s) to be used for extraction.

Interposer Flow

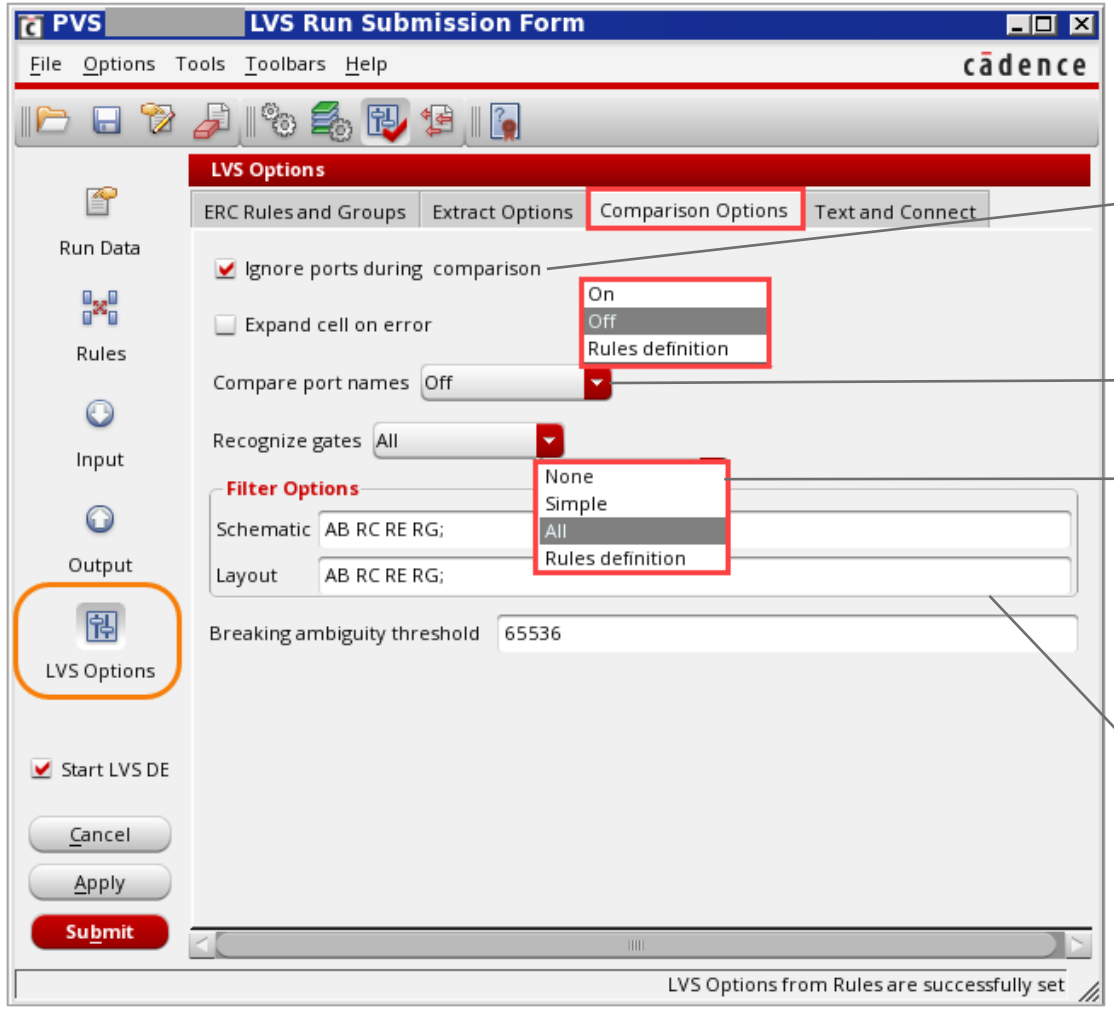
- Performs 3D-aware wire interconnect check on cell.
- Supports only one cell (first declaration).
- Not be used in the PVS-Quantus flow.

- LVS to find shorts that can be isolated.
- Creates flat-extracted netlist file.
- Count the number of shapes that overlap stamped layer for each net participated in the conflict and selects one with largest number of shapes.

LVS Run Submission Form

LVS Options – Comparison Options

PVS LVS Run Submission Form → LVS Options → Comparison Options



- Ignores ports: Does not compare the names of the nets on the ports.
- Expands cells with following kind of errors: Reduce, Parameter, Pins issues.

- Compare Port Names: Compares discrepancies between matching pins, layout and schematic.

Should SVS identify logic gates during comparison?

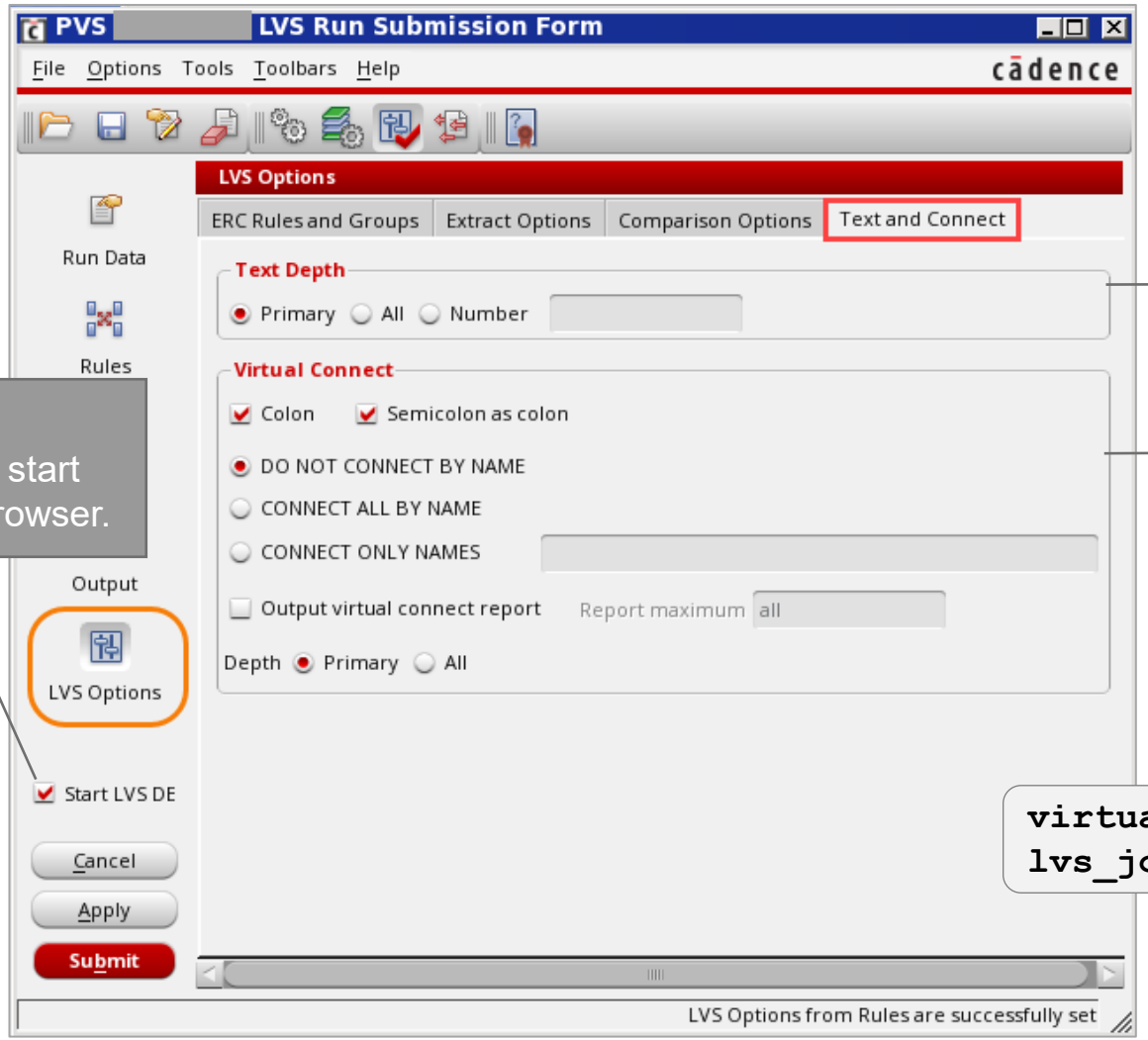
- Rules Definition: Default. Use Rule deck settings.
- None: Does not identify.
- Simple: NOT, AND, OR, NOR, NAND, XOR, XNOR.
- All: Every logic gate combination identified.

- Filter out (lvs_filter_option) specific configurations from Schematic and/or layout.
- E.g., lvs_filter_option AB –source.

LVS Run Submission Form

LVS Options – Text and Connect

PVS LVS Run Submission Form → LVS Options → Text and Connect



Start LVS DE
- Default. To start the error browser.

Text Depth:
- PRIMARY: Top level only. Default.
- ALL levels.
- NUMBER: Specific level only.

Virtual text connections:
- Instruct LVS tool to connect nets ONLY for verification runs. No physical connections.
- Colon connect: Texts are named with a ":" or ";"
- Additional options to:
  - Generate a report.
  - Define Depth.
- **Note**: Remove virtual connect definitions before signoff LVS run to avoid masking actual opens.

```
virtual_connect -colon yes -depth 3 -name VDD VCC
lvs_join_nets cellA VDD VCC vdc -layout
```

You are all set to start LVS run; click **Apply** / **Submit**.

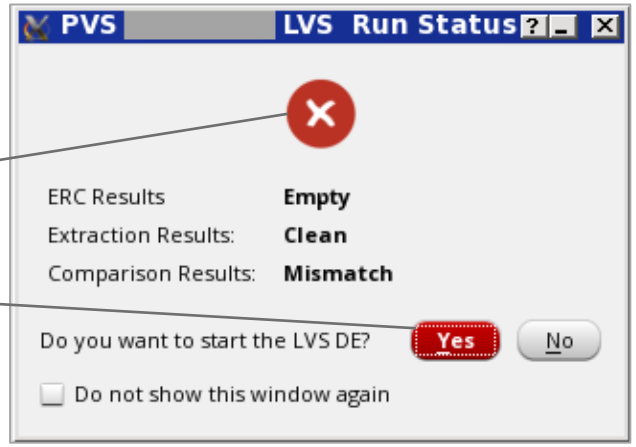
LVS Debug Environment (LVSDE)

Introduction

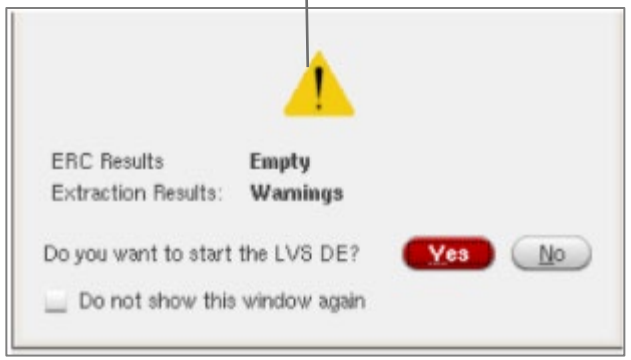
LVS Debug Environment (DE) → Shorts / Comparison Tabs

Errors and/or mismatch

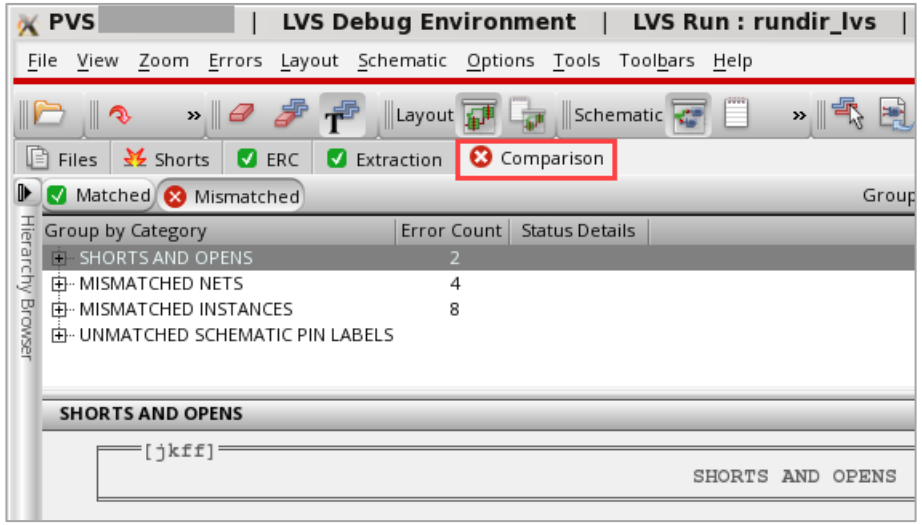
Click **Yes** to start PVS LVS DE



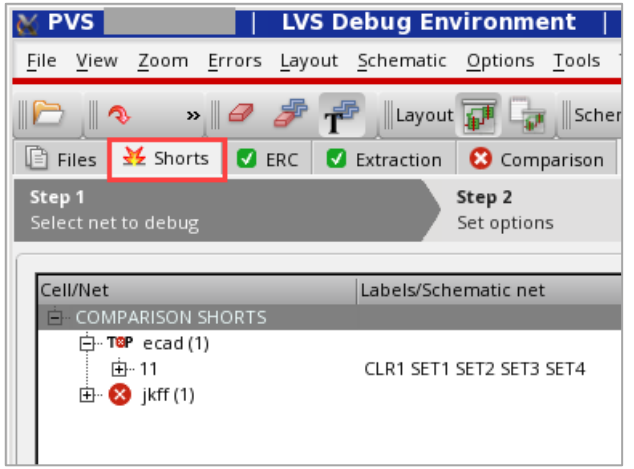
Only Warnings exist



Run results are clean



- Tabs in LVS DE:
  - Files
  - Shorts
  - Sconflicts
  - ERC results
  - Extraction results
  - Comparison results



** Shorts/Sconflicts tabs ONLY if corresponding errors exist in the design

LVS Error Types

Typical errors encountered during LVS include:

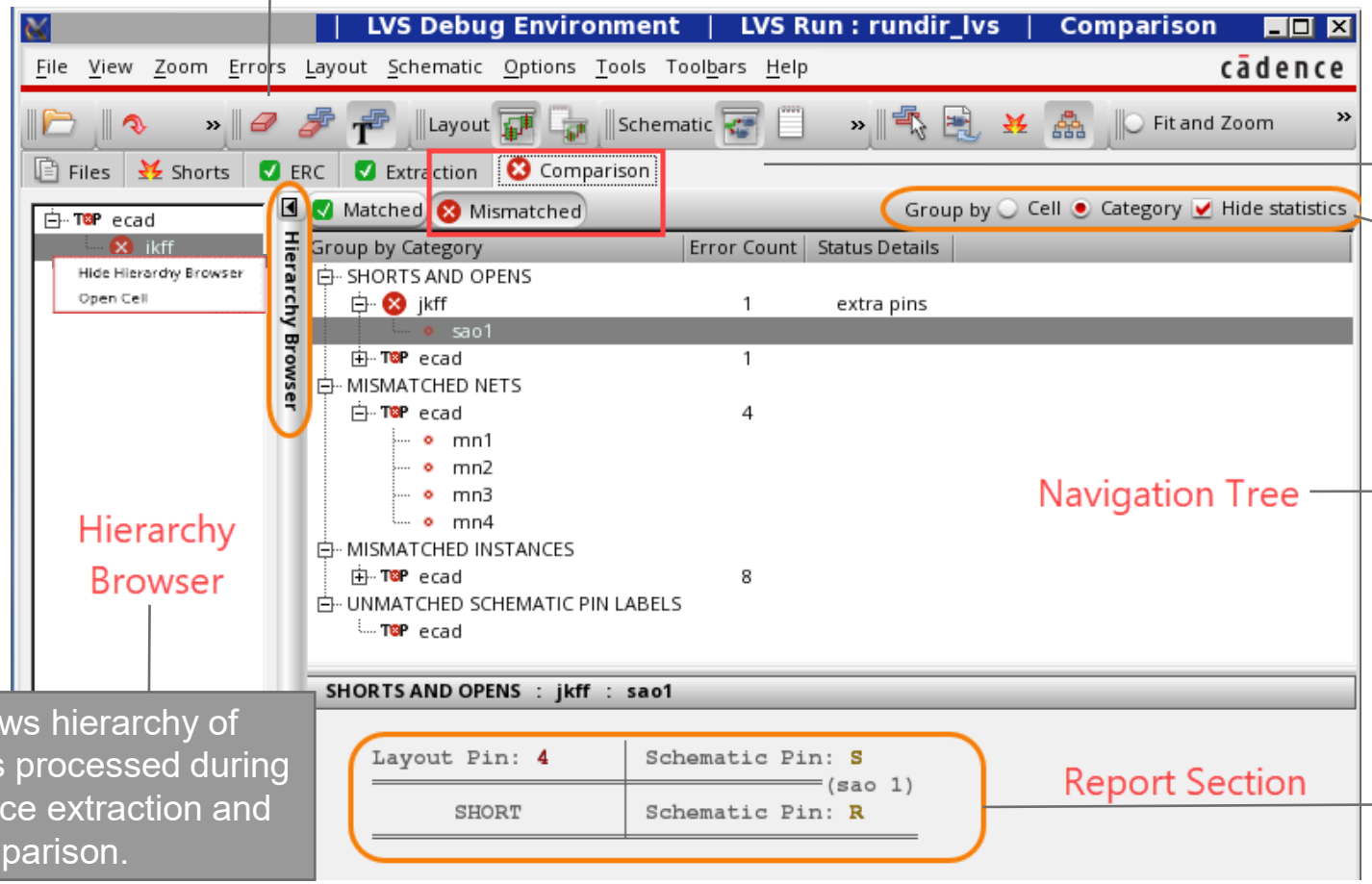
- Shorts: Two or more wires that should not be connected have been and must be separated.
- Opens: Wires or components that should be connected are left dangling or only partially connected. These must be connected properly to fix this.
- Component Mismatches: Components of an incorrect type have been used.
  - Example: A low Vt MOS device instead of a standard Vt MOS device.
- Missing Components: An expected component has been left out of the layout.
- Parameter Mismatch: Components in the netlist can contain properties. The LVS tool can be configured to compare these properties to a desired tolerance. If this tolerance is not met, then the LVS run is deemed to have a Property Error.
  - Example: if a resistor in a schematic had resistance=1000 (ohms) and the extracted netlist had the matched resistor with resistance=997(ohms) and the tolerance was set to 2%, then this device parameter would pass as 997 and is within 2% of 1000.
  - 997 is 99.7% of 1000 which is within the 98% to 102% range of the acceptable +-2% tolerance error.

LVS Debug Environment

Comparison Tab

LVS Debug Environment (DE) → Comparison Tab

Clear highlights in layout or schematic viewers.



Shows hierarchy of cells processed during device extraction and comparison.

Comparison Mismatched tab list Mismatched nets, pins and devices plus shorts and opens.

Multiple options to list the error items.

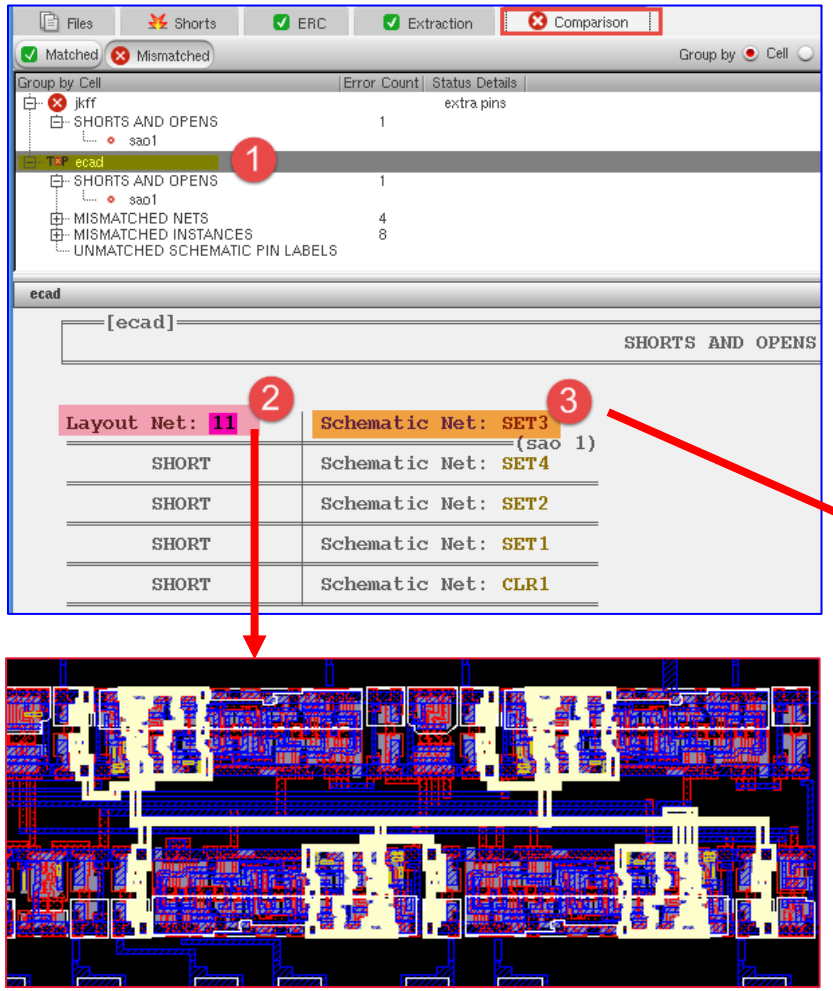
Navigation tree represents report file generated by comparison (<runName.rep.cls>).

Report section:

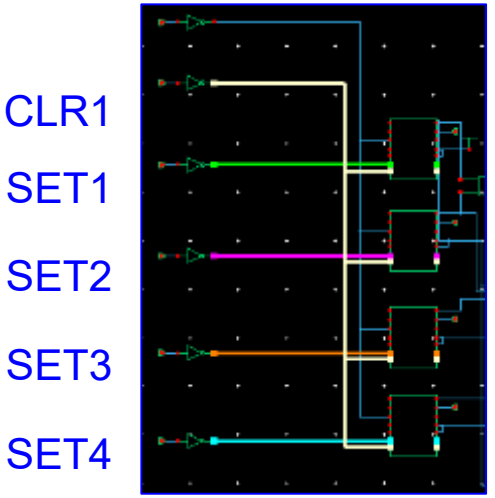
- Displays part of comparison report relates to selected item in the Navigation tree. ALL levels.
- Messages are hypertexted. Select any link to highlight and zoom to net/instance in the viewer.

How to Analyze Comparison Mismatches in LVS DE?

LVS Debug Environment (DE) → Comparison Tab → Mismatched



1. Click on a cell name involved in the violation.
2. Click on layout Net 11: and it opens and highlights net11 in the layout.
3. Click on schematic nets: They are links to open the schematic and highlight chosen nets.

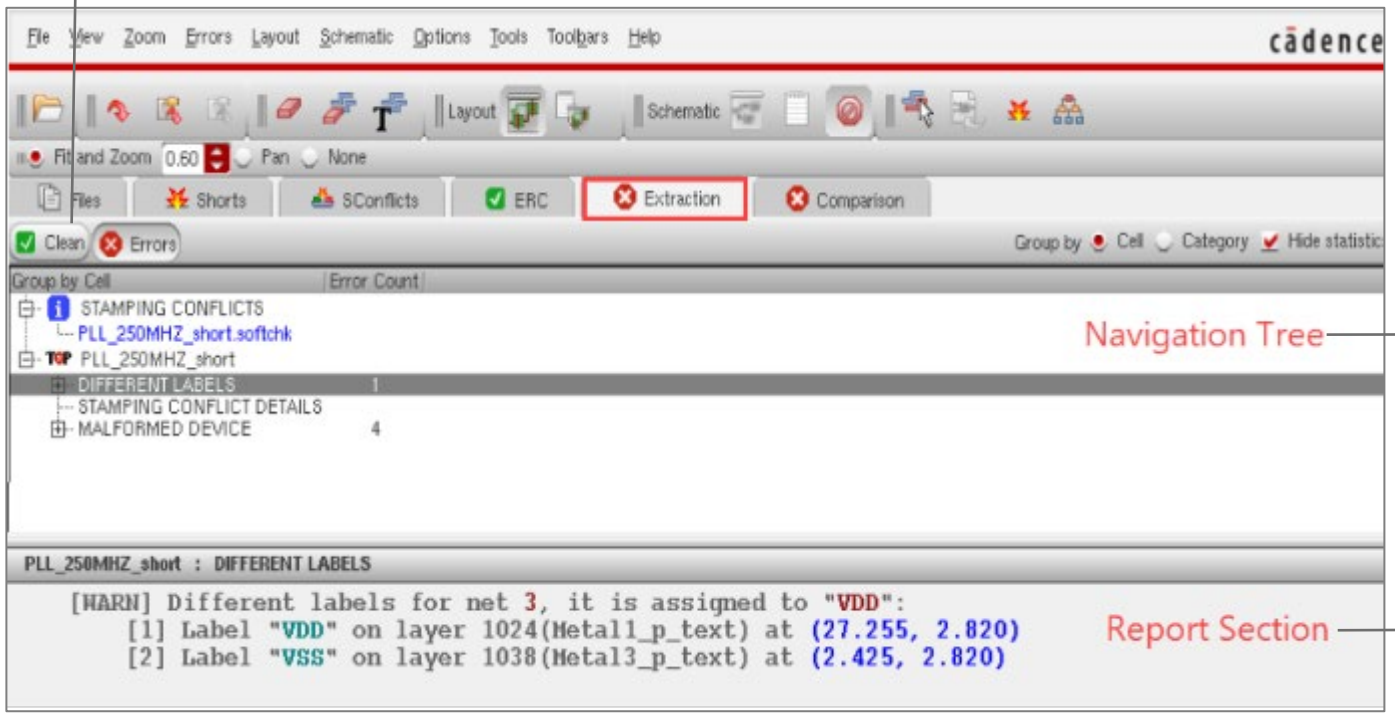


LVS Debug Environment

Extraction Tab

LVS Debug Environment (DE) → Extraction Tab

Clean indicates no technical errors for this LVS run.



Navigation Tree lists errors/warnings being found in the design by device extraction.

Report Section:

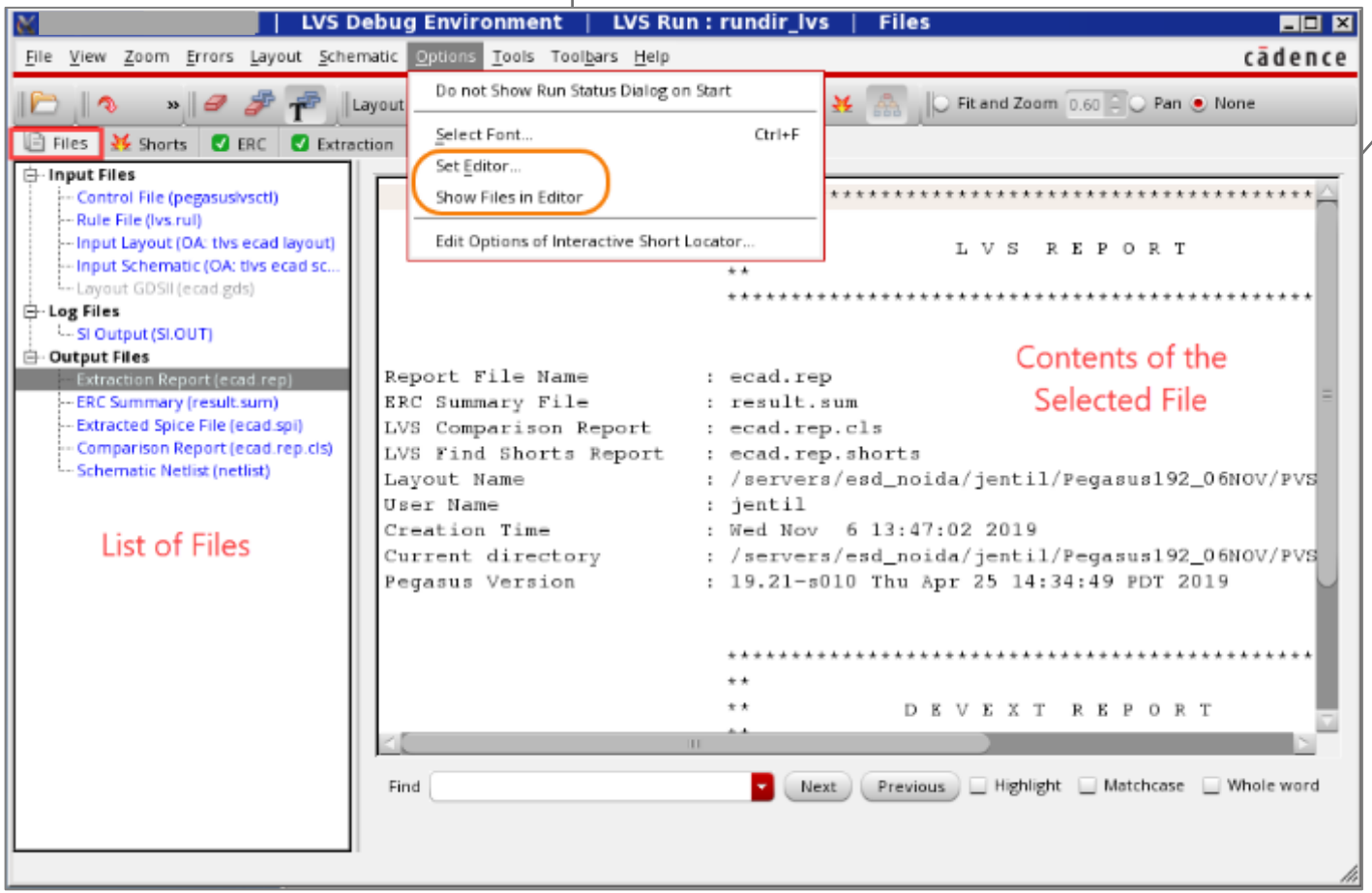
- Displays part of device extraction report relates to selected item in the Navigation tree. ALL levels.
- Messages are hypertexted. Select any link to highlight and zoom to net/instance in the viewer.

LVS Debug Environment

Files Tab

LVS Debug Environment (DE) → Files Tab

Set and open any file in external editor.



Contents of the selected file are shown in the viewer in the right-hand window.

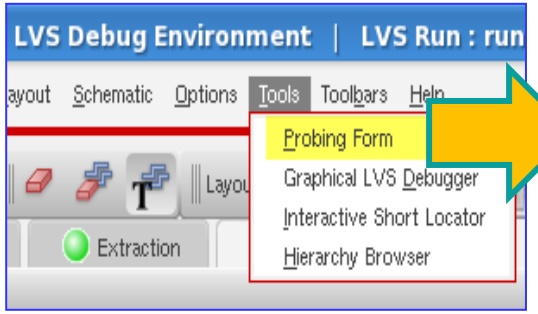
Output File Name	Description
ASCII Files: [report files that you can read]	
reportName.erc.sum	ERC summary report.
reportName.rep	Extraction report file.
reportName.rep.cls	Comparison report file.
reportName.cxl	Comparison cross-reference file.
reportName.cps	Pin swap analysis file.
reportName.cfr	Filtered devices file.
Layout and Schematic Netlist Files	
cellName.spi	Extracted SPICE file.
cellName.cdl	Schematic netlist.

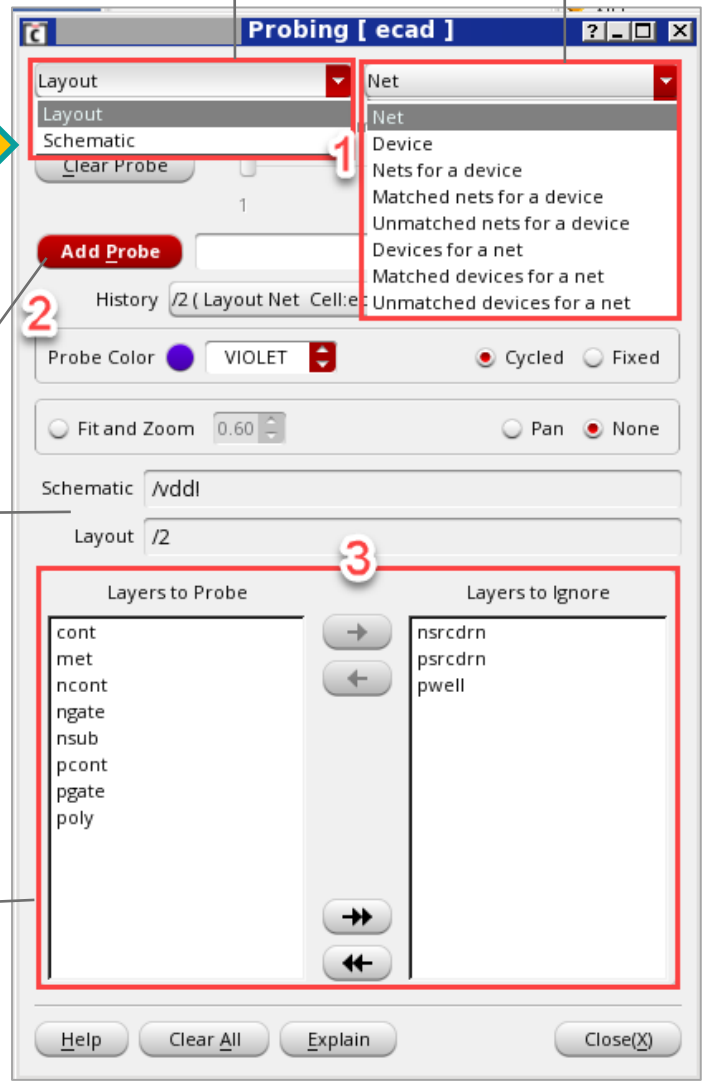
How to Probe Nets in the LVS Error Browser?

The Probing form lets you view the specified layout errors in a layout editor window, such as Virtuoso.

PVS LVS Debug Environment → Tools → Probing Form

1. Make Schematic or Layout selection for probing.
2. Add Probe by specifying a net.
3. Select the Layers to Probe and Ignore.





What Is the Interactive Short Locator (ISL)?

ISL is an LVS shorts debug tool which identifies nets containing two or more labels that are not the same. ISL employs the *One-pass Short Isolation* technique to display the paths that consists of shapes from the connected layers between the shorted labels.

- Recommendation: Debug shorts first since it affects results in other tabs like extraction & comparison.
  - Short happens when two or more nets in the netlist are connected together incorrectly.
  - The Shorts tab lists shorts found in the design. This tab is hidden If there are no shorts in design.
- ISL is supported in Virtuoso/Innovus™.
- In GUI mode, ISL performs first run when initialized from shorts tab in LVSDE & then proceed to debug.
- Option to save / load state (current analysis results) of Interactive Short Locator.
- ISL features for short analysis:
  - Interactive navigation : of the shapes along the shorted path
  - Interactive labelling : interactively place additional labels to reduce the path results
  - Interactive analysis : of suspicious shape for short
  - Multithreading : reduces time needed for short analysis

How to Set Up the Interactive Short Locator (ISL) Options?

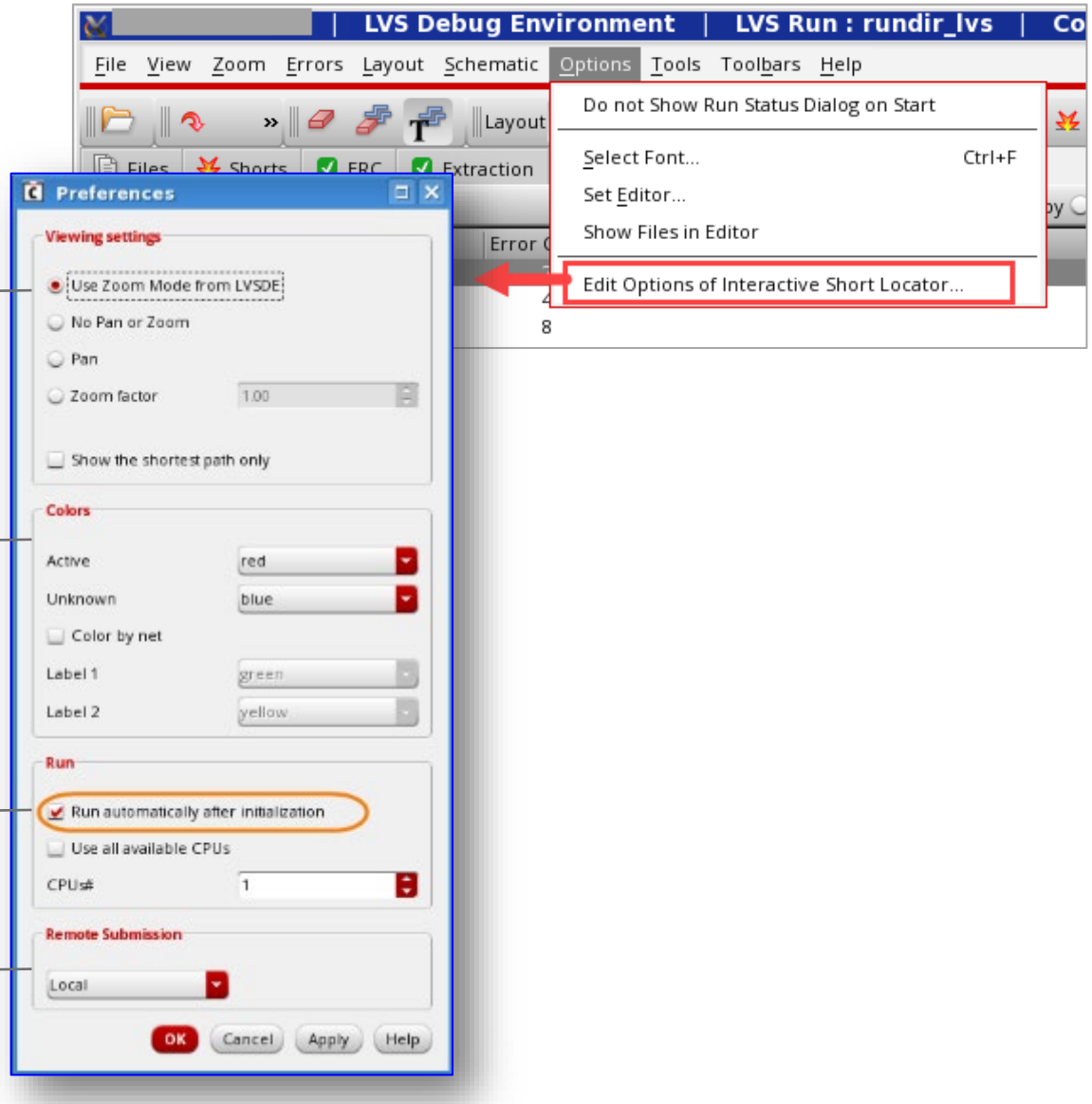
PVS LVSDE → Options → Edit Options of Interactive Short Locator → ISL Preferences form

Use Zoom Mode from LVSDE – Keeps the same zoom settings as set for the LVSDE browser.

Colors shown here come from the LVS_COLORS file or the default colors when no file is found.

Run automatically after initialization – (default) Starts to run automatically after the initialization of ISL.

Remote Submission options: Local, RSH, SSH, LSF.



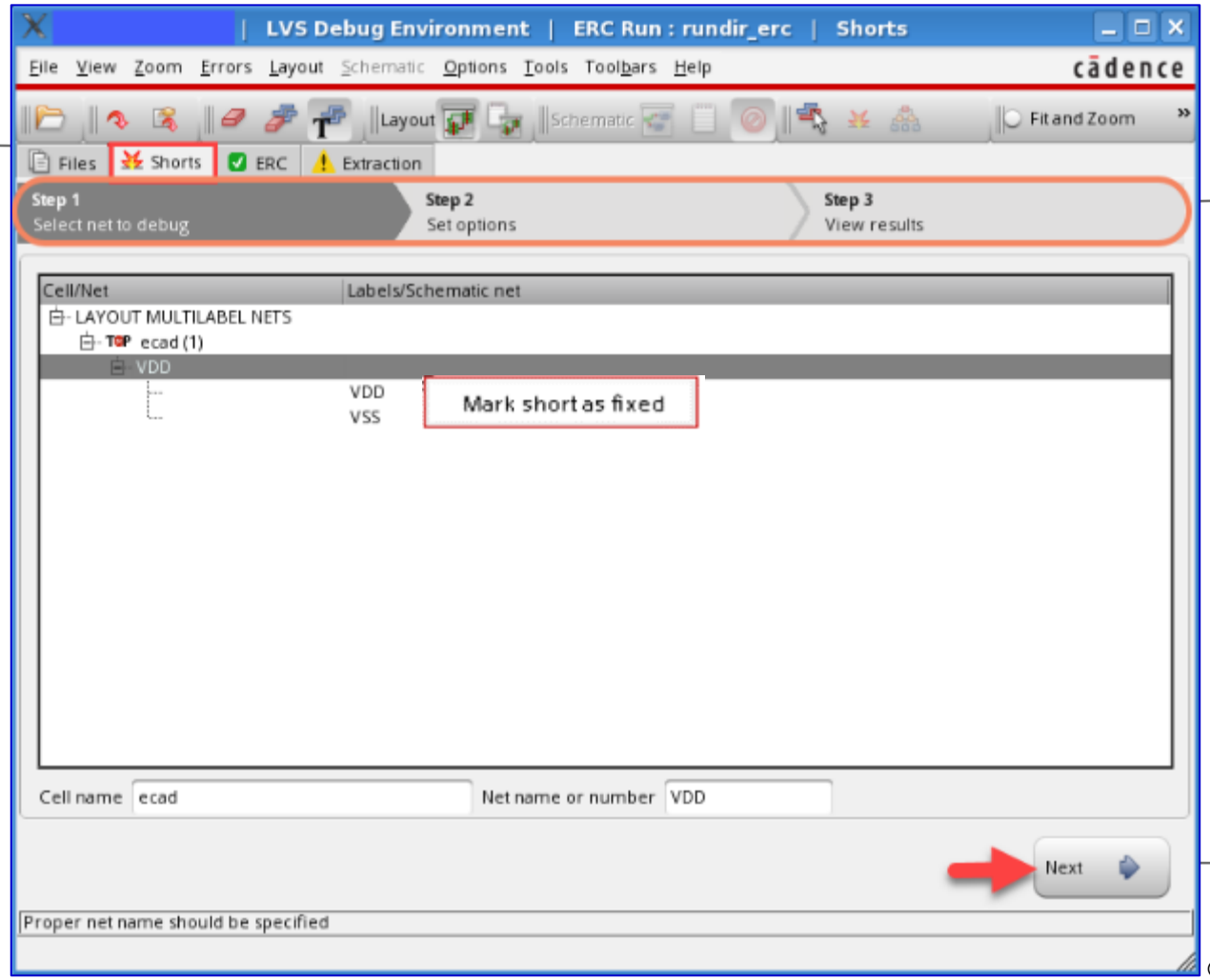
How to Debug Shorts in LVS DE Using ISL?

Shorts happen when two or more nets in the netlist have been connected together incorrectly. To debug shorts in LVS/ERC Error Browsers, PVS LVS uses a feature called Interactive Short Locator (ISL) that identifies a net containing two or more labels that are not the same.

Steps 1 & 2: Go to the Shorts Tab in LVS DE and Select Nets

Shorts Tab in LVSDE | Select Nets to Debug | Set Debug Options | Initialize & Run ISL | Analyze ISL Results | Add Labels | Add Split Box | Select Layers

The Shorts tab appears and lists shorts if found in the design.



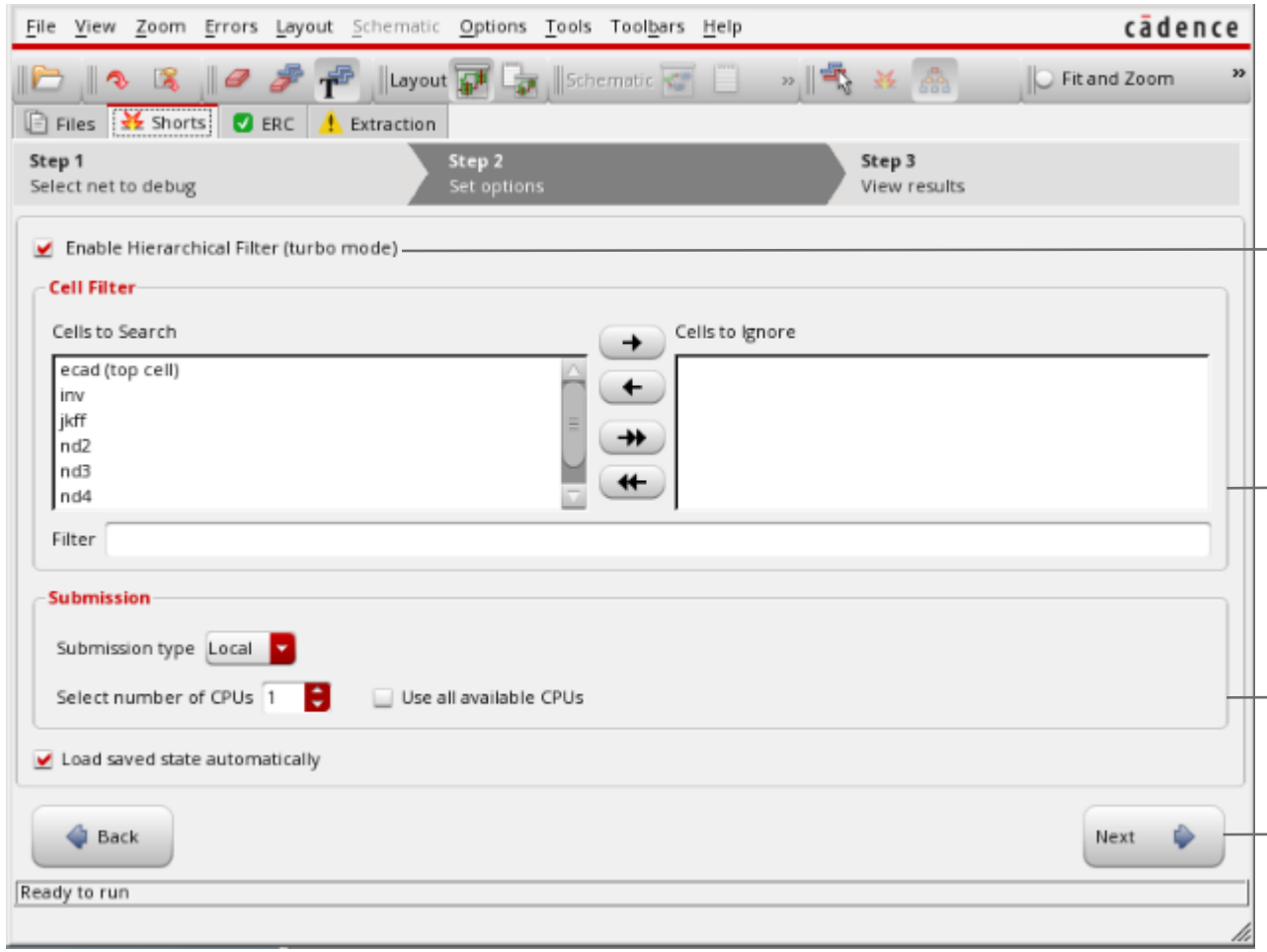
There are three steps in the shorts debug process:

1. Select net to debug.
2. Set options.
3. View results.

- Click **Next** to go to step 2 – Set options.

Step 3: Set Options for Shorts Debug

Shorts Tab in LVSDE → Select Nets to Debug → **Set Debug Options** → Initialize & Run ISL → Analyze ISL Results → Add Labels → Add Split Box → Select Layers



Enable Hierarchical Filter (turbo mode) (recommended):

- By default, ISL checks all levels of hierarchy for shorts → Large memory usage and runtime.
- Turbo mode checks hierarchies top-down in batches.
- If shorts are found, ISL stops the job and display results.
- Saves time and memory.
- Ideal use in large designs where lower blocks are clean.

Cell Filter

- Select a subset of cells to be used for analysis.

To run ISL in distributed processing across multi-CPUs.

Click **Next** to go to step 3 – View Results.

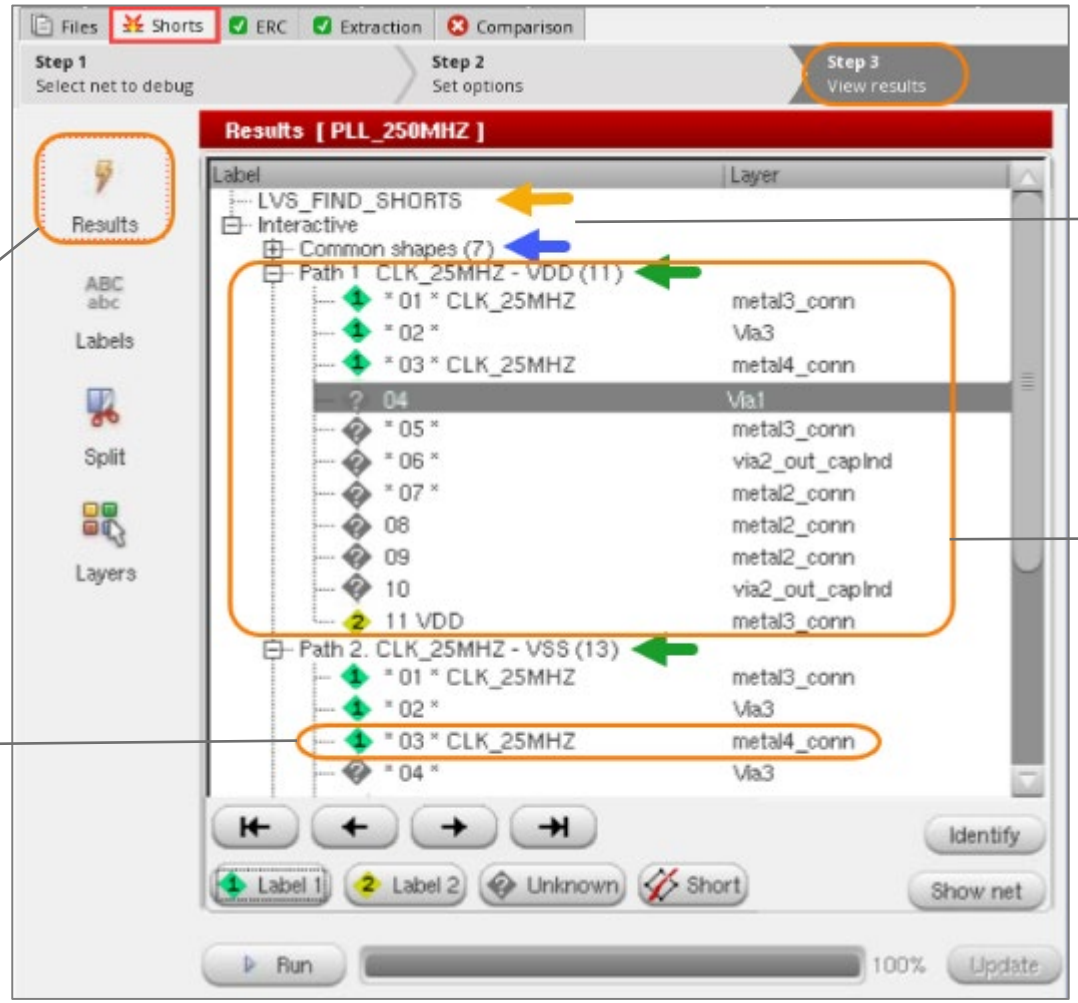
Step 4: Initialize and Run ISL

Shorts Tab in LVSDE → Select Nets to Debug → Set Debug Options → **Initialize & Run ISL** → Analyze ISL Results → Add Labels → Add Split Box → Select Layers

In step 3, first round of ISL is run by default and results are displayed on the Results panel.

The Results panel facilitates:
- Displaying/Navigating errors.
- Marking shapes as part of selected net.
- Rerunning/stopping ISL engine.

- Each shape comes with an indicator, sequence #, label & layer info.



- Results path from LVS_FIND_SHORTS & ISL engine are displayed.
- Common shapes are markers common for both (recommended to clean first).

- ISL shorts are listed under Interactive first level.
- Proceed to the Path results after Common shapes.
- Known locations of Label 1 & Label 2 are established as start/end keys of shorts path.

Step 5: Analyze ISL Results

Shorts Tab in LVSDE → Select Nets to Debug → Set Debug Options → Initialize & Run ISL → **Analyze ISL Results** → Add Labels → Add Split Box → Select Layers

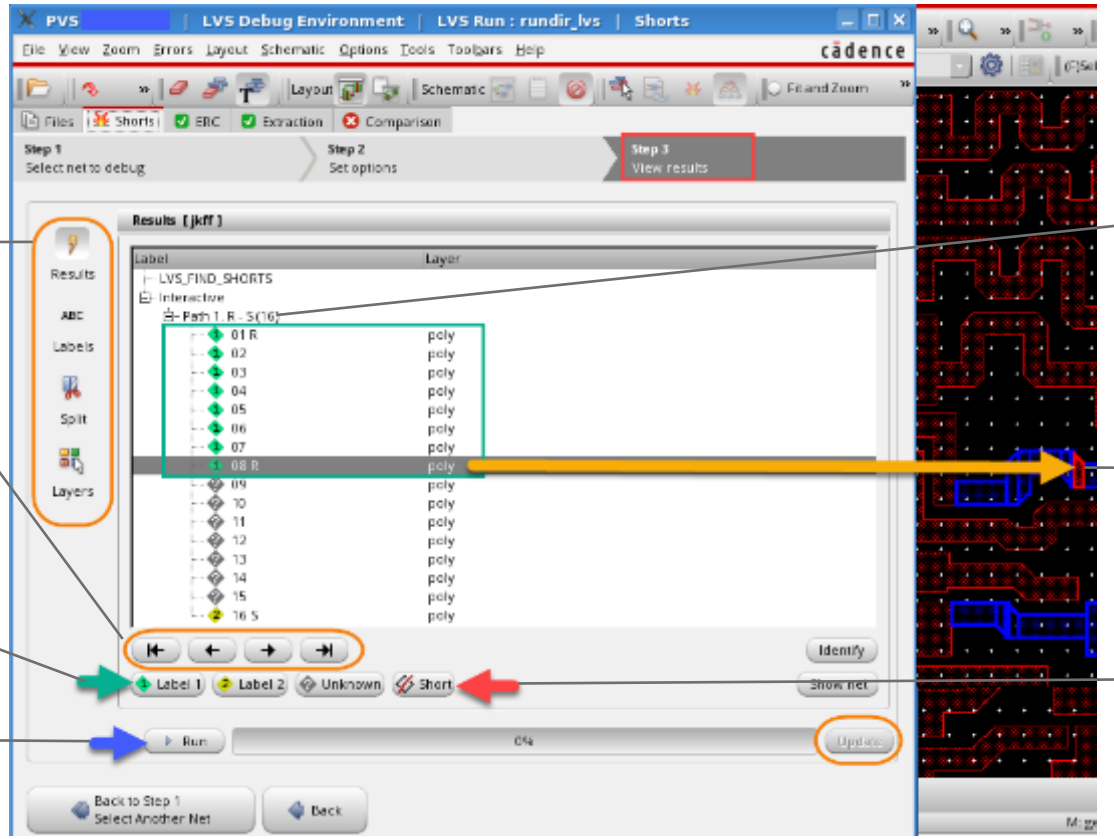
Short isolation process: Traverse the path, add virtual labels to shapes in the path and rerun ISL (no need to rerun LVS extraction ☺).

3. Features in sidebar panels can be used interactively to reduce shapes count in the paths.

4. Use the arrow (→, |←) keys, identify and show net options to navigate shapes.

5. Each shape in the path can be virtually labelled as Label1, Label2, Unknown or Short**.

6. Rerun ISL to narrow down shorts list. Repeat the steps till design is clean.



1. Select ISL interactive Path 1. R-S (16) with 16 shapes – the path is highlighted in the layout.

2. Selected shape(s) in the path get highlighted in a different color.

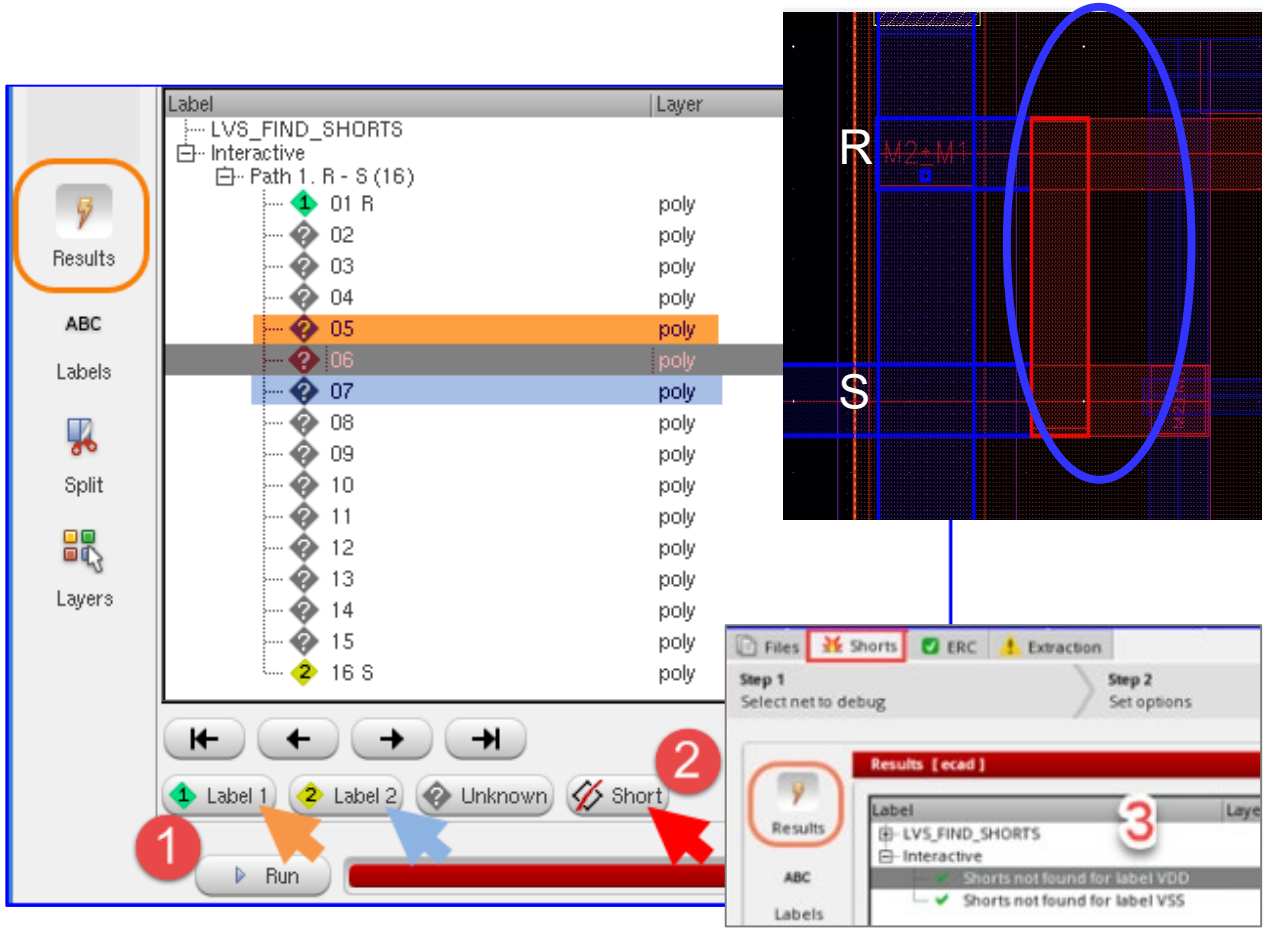
**Mark As Short: "What if" analysis: Virtually marks selected shape as “short” and excluded during next run. If result is clean, remove shape from layout and rerun.

Step 5: Analyze ISL Results

Short Isolation: Example

Let us assume that shape #6 is suspected to be shorting R and S nets. Then options are:

1. Assign Label1 to #5, and Label2 to #7 and rerun ISL (and/or).
2. Assign Short status to #6 and rerun ISL.
3. If #6 is indeed the shorting shape, the run comes out CLEAN. Go ahead and remove that shape in the layout. Rerun LVS.



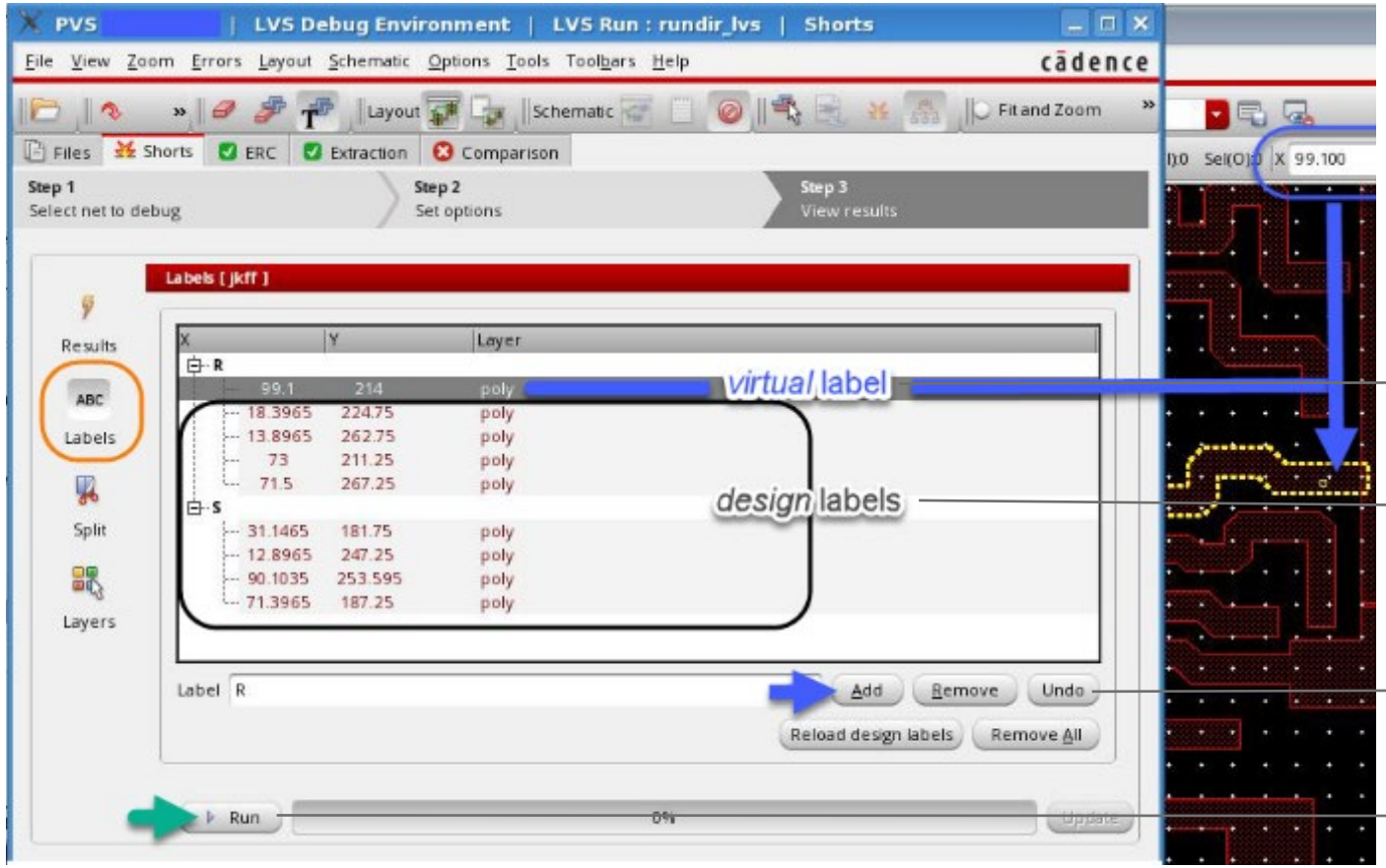
ISL for Shorts Between Unlabeled Nets:

- LVS_FIND_SHORTS will not find such shorts. Reported only in comparison reports.
- Select in COMPARISON SHORTS of Init panel to initialize ISL.
- ISL places labels automatically and generate paths between shorted nets.

Step 6: Virtually Label Nets & Rerun ISL

View Results – Labels

Shorts Tab in LVSDE → Select Nets to Debug → Set Options → Run ISL → Analyze ISL Results → **Add Labels** → Add Split Box → Select Layers

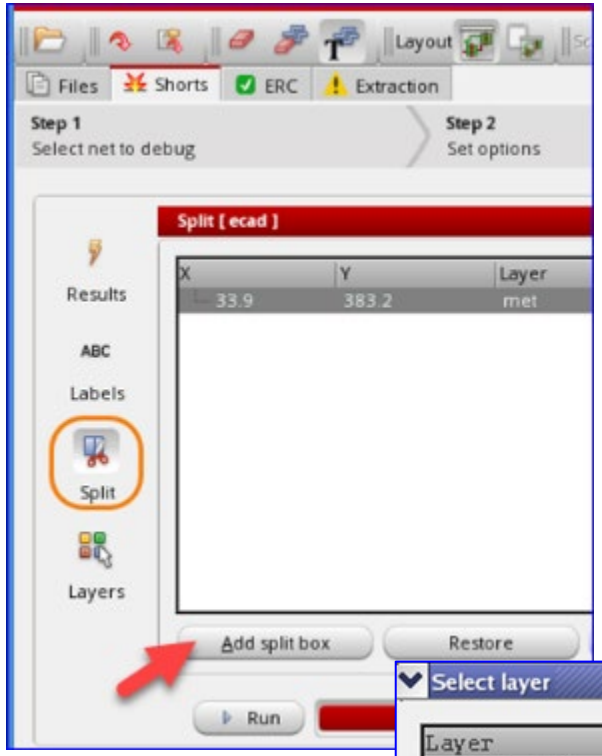


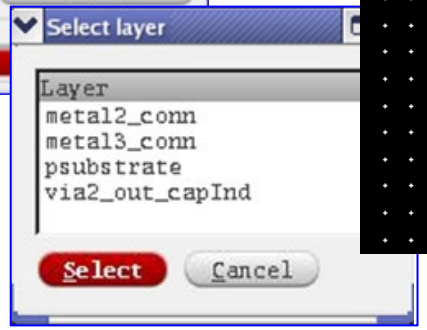
- The Labels panel manages the labels (design & virtual) used by the ISL engine.
- In layout, design labels are highlighted in **black** & virtual labels in blue.

- Virtual labels are added through options in Results (Label1, Label2) Labels panel.
- ISL engine loads relevant design labels.
- Labels with the same name will be grouped.
- Option to Add, Remove, Undo virtual labels and Restore design labels anytime.
- Rerun ISL with new labels.

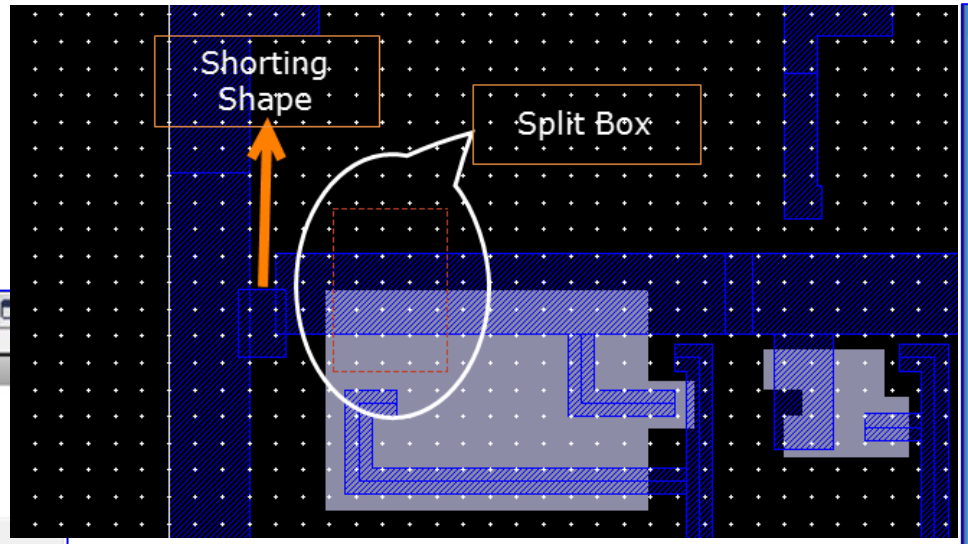
Step 7: Add Split Box

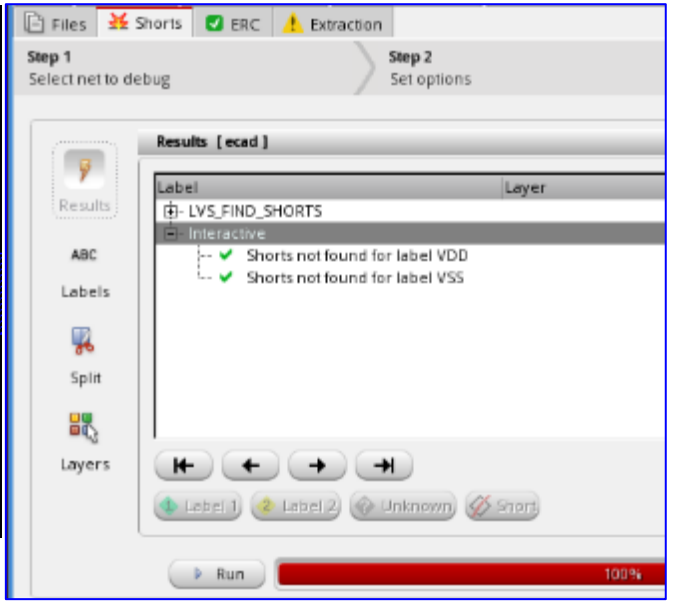
View Results – Split





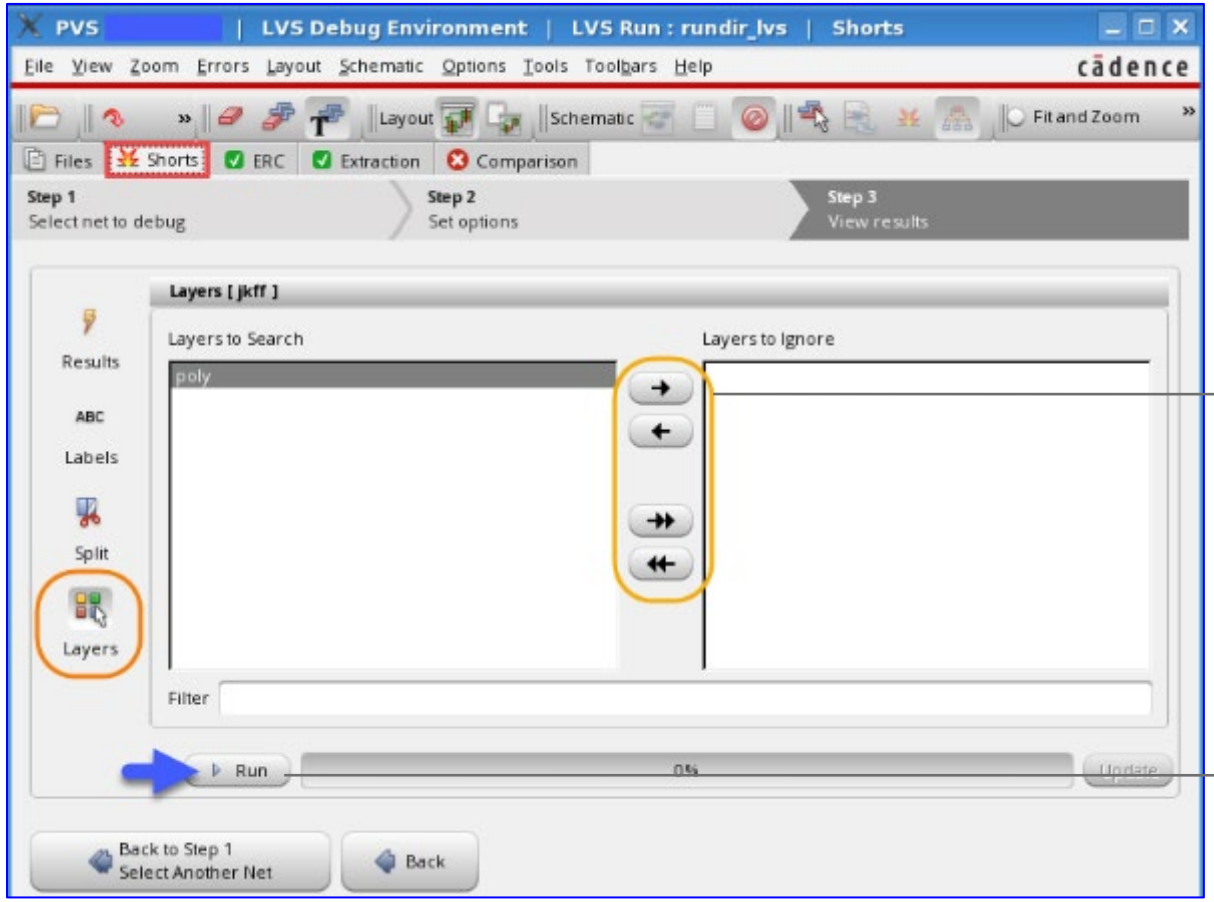
- Split panel provides an alternative way to debug short.
  1. Introduce a Split Box at the path.
  2. Select the area to exclude and layer(s) to split (supports multi-layer split).
  3. Virtually isolate problematic shape and rerun ISL.
  4. If the run is clean, remove that shape and rerun LVS.





Step 8: Select Layers for ISL

View Results – Layers



- By default, ISL uses all connectivity layers during analysis.

- The Layers panel (Layers to Search & Ignore) let the users select/ignore a subset of layers for ISL analysis.
- Layers used in the CONNECT and SCONNECT rules of the extraction rule deck are listed here.

Caution

- If short(s) are involved in the layers removed from analysis, that may lead to incorrect analysis.

- Use the Run button to rerun ISL at any stage/panel.

LVS: Frequently Asked Questions

What do you do if LVS gets stuck at sconnect/connect?

- Check the rule deck for **`text_depth`**.

```
TEXT_DEPTH -PRIMARY;
PORT -DEPTH -PRIMARY;
```

- Make sure stdcell texts are not promoted to top.

What if LVS is stuck at mismatch due to physically unconnected nets?

- You need to short them in the rule deck virtually for LVS.
- Schematic Side: **`lvs_join_nets`**:

```
lvs_join_nets cellA VSS1  VSS2 -source          // joins nets VSS1 and VSS2 on schematic side
lvs_join_nets cellA VDD  VCC -source            // joins nets VDD and VCC on schematic side
```

  - Note*:* Must remove **`lvs_join_nets`** before final LVS, otherwise it might mask the open connections.
- Layout Side: To manage open connections on the layout side.
  - Typically, nets that are meant to connect at higher levels of design.

```
virtual_connect -colon yes;             // connects labels with identical names ending in a colon (:)
virtual_connect -semicolon yes;         // connects labels with identical names ending a semi-colon (;)
virtual_connect -name vdd vcc;          // connects nets vdd and vcc
virtual_connect -name vdd?              //connects vdd1, vdd2, vdd3, vdda, vddb .. ..
```

cadence®