

Incisive Metrics Center User Guide

Product Version 15.20

February 2016

© 2016 Cadence Design Systems, Inc. All rights reserved.

Printed in the United States of America.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. (Cadence) contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522.

This product contains third party software. Refer to *<install_dir>/doc/thirdpartyinfo/IMCthirdpartyinfo.txt* for details on copyright and licensing terms.

Open SystemC, Open SystemC Initiative, OSCI, SystemC, and SystemC Initiative are trademarks or registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries and are used with permission.

All other trademarks are the property of their respective holders.

Restricted Permission: This publication is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this publication, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used only in accordance with a written agreement between Cadence and its customer.
2. The publication may not be modified in any way.
3. Any authorized copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.
4. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Contents

1

<u>Getting Started with Incisive Metrics Center</u>	11
<u>1.1 About IMC</u>	11
<u>1.2 Quick Start Task Flow</u>	12
<u>1.3 Launching IMC</u>	13
<u>1.3.1 Launching IMC in 32-Bit Mode</u>	15
<u>1.3.2 Launching IMC from Enterprise Manager</u>	16
<u>1.3.3 Launching IMC from SimVision</u>	16
<u>1.4 IMC Interface</u>	16
<u>1.4.1 Menu Bar</u>	17
<u>1.4.2 Toolbar</u>	26
<u>1.4.3 Navigation Pages</u>	36
<u>1.4.4 Coverage View Area</u>	38
<u>1.4.5 Messages Log</u>	38
<u>1.5 Loading Runs</u>	39
<u>1.5.1 Reloading Run</u>	42
<u>1.5.2 Unloading Run</u>	43
<u>1.6 Metrics Page</u>	43
<u>1.6.1 Verification Hierarchy Pane</u>	44
<u>1.6.2 List tabs Pane</u>	58
<u>1.6.3 Details Pane</u>	60
<u>1.7 Managing Table Columns</u>	64
<u>1.7.1 Resizing Columns</u>	64
<u>1.7.2 Reordering Columns</u>	64
<u>1.7.3 Sorting Column's Values</u>	64
<u>1.7.4 Removing Columns</u>	64
<u>1.7.5 Adding Columns</u>	64
<u>1.8 Filtering Table Data</u>	66
<u>1.8.1 Filtering Textual Values</u>	66
<u>1.8.2 Filtering Numerical Values</u>	67
<u>1.8.3 Filtering Values in Ex Column</u>	68

<u>1.8.4 Advanced Filtering</u>	70
1.9 Defining and Organizing Views	72
<u>1.9.1 Defining Views</u>	73
<u>1.9.2 Saving Views</u>	73
<u>1.9.3 Deleting Views</u>	74
<u>1.9.4 Renaming Views</u>	75
<u>1.9.5 Setting a Default View</u>	75
<u>1.9.6 Organizing Views Under Different Folders</u>	76
1.10 Exporting Metrics Tree to a CSV File	77
1.11 Configuring Configurable Items	78
<u>1.11.1 Configuring General Options</u>	79
<u>1.11.2 Configuring Appearance Options</u>	80
<u>1.11.3 Configuring Refinement Options</u>	81
<u>1.11.4 Configuring Messages Bundling Options</u>	83
<u>1.11.5 Configuring Formal Analysis Options</u>	84
<u>1.11.6 Configuring Merge Options</u>	84
<u>1.11.7 Configuring Parallel Merge Options</u>	86
<u>1.11.8 Configuring Parallel Ranking Options</u>	99
<u>1.11.9 Configuring Parallel Unique Coverage Report Options</u>	101
2	
Analyzing Metrics Data	105
2.1 Code Coverage	105
<u>2.1.1 Block Coverage</u>	105
<u>2.1.2 Expression Coverage</u>	115
<u>2.1.3 Toggle Coverage</u>	124
2.2 FSM Coverage	132
<u>2.2.1 Identify State Machine for FSM Coverage Analysis</u>	132
<u>2.2.2 Launch the FSM Page</u>	132
2.3 Functional Coverage	144
<u>2.3.1 Assertion Coverage</u>	144
<u>2.3.2 Covergroup Coverage</u>	153

3

<u>Refining Metrics Data</u>	167
<u>3.1 Refinement in IMC GUI</u>	167
<u>3.1.1 Excluding Instances or Types</u>	168
<u>3.1.2 Excluding Specific Metrics</u>	170
<u>3.1.3 Excluding Specific Items</u>	171
<u>3.1.4 Excluding Connected Entities (Smart Refinement)</u>	176
<u>3.1.5 Adding Comments at the Time of Exclusion</u>	180
<u>3.1.6 Editing User Comments</u>	183
<u>3.1.7 Hide Excluded Items</u>	184
<u>3.1.8 Un-Excluding Excluded Items</u>	185
<u>3.1.9 Saving Refinements</u>	186
<u>3.1.10 Loading Saved Refinements</u>	187
<u>3.2 Refinement in Command-Line Interactive Mode</u>	188
<u>3.2.1 Command Syntax</u>	188
<u>3.2.2 Excluding All Coverage Objects in an Instance or Type</u>	191
<u>3.2.3 Excluding Blocks</u>	192
<u>3.2.4 Excluding Expressions</u>	194
<u>3.2.5 Excluding Toggle Signals</u>	198
<u>3.2.6 Excluding FSMs</u>	205
<u>3.2.7 Excluding Assertions</u>	212
<u>3.2.8 Excluding Covergroup Items</u>	214
<u>3.2.9 Excluding Unreachable Items</u>	221
<u>3.2.10 Un-Excluding Excluded Items</u>	223
<u>3.2.11 Saving Refinements</u>	224
<u>3.2.12 Loading Saved Refinements</u>	225
<u>3.3 Reusing Refinements File</u>	226
<u>3.4 Converting an ICF to an IMC Exclude File</u>	232

4

<u>IMC Command-Line Interface</u>	237
<u>4.1 Launching IMC</u>	237
<u>4.1.1 Command Syntax</u>	238
<u>4.2 Loading Runs</u>	243

Incisive Metrics Center User Guide

<u>4.2.1 Listing Currently Loaded Run</u>	244
<u>4.2.2 Unloading Run</u>	244
<u>4.3 Merging Runs</u>	244
<u>4.4 Ranking Runs</u>	244
<u>4.5 Generating Reports</u>	245
<u>4.6 Refining Metrics Data</u>	245
<u>4.6.1 Reusing Refinements</u>	245
<u>4.6.2 Loading and Applying Refinements</u>	245
<u>4.6.3 Saving Refinements to a Single File</u>	246
<u>4.7 Exporting Metrics Data to CSV File</u>	246
<u>4.8 Mapping Source Path</u>	247
<u>4.9 Setting/Removing Alias for Commands</u>	248
<u>4.9.1 Setting an alias (alias)</u>	248
<u>4.9.2 Removing an alias (unalias)</u>	249
<u>4.10 Setting/Viewing Preferences</u>	249
<u>4.10.1 Viewing Preferences</u>	250
<u>4.10.2 Setting Preferences</u>	251
<u>4.11 Configuring Configurable Items</u>	252
<u>4.12 Viewing History Information</u>	255
<u>4.13 Viewing Help</u>	255
<u>4.14 Viewing Performance Information</u>	255
<u>4.15 Launching Documentation</u>	256
<u>4.16 Exiting IMC</u>	256
<u>4.17 Using Environment Variables in CLI Commands</u>	256
<u>4.18 Commands Availability with Incisive Enterprise Simulator License</u>	257

5

<u>Merging Runs</u>	259
<u>5.1 Merging Data</u>	259
<u>5.1.1 Command Syntax</u>	259
<u>5.1.2 Merged Data Storage</u>	264
<u>5.2 Specifying Merge Configuration</u>	267
<u>5.2.1 Specify the DUTs for Merge</u>	267
<u>5.2.2 Set the Mode of Merge in Cross-Hierarchy Merging</u>	272
<u>5.2.3 Role of -resolution Union in Creation of Resultant Model</u>	284

<u>5.3 Precedence while Merging</u>	295
---	-----

6

<u>Ranking Runs</u>	297
---------------------------	-----

<u>6.1 Setting the Ranking Configuration</u>	297
--	-----

<u>6.1.1 Command Syntax</u>	297
-----------------------------------	-----

<u>6.2 Ranking the Runs</u>	299
-----------------------------------	-----

<u>6.2.1 Command Syntax</u>	299
-----------------------------------	-----

<u>6.2.2 Ranking Examples (Text Format)</u>	303
---	-----

<u>6.2.3 Ranking Examples (HTML Format)</u>	311
---	-----

7

<u>Reporting and Grading</u>	315
------------------------------------	-----

<u>7.1 Generating Reports in GUI</u>	315
--	-----

<u>7.2 Generating Reports in Interactive Command-line Mode</u>	327
--	-----

<u>7.2.1 The report Command</u>	327
---------------------------------------	-----

<u>7.2.2 The report_metrics Command</u>	391
---	-----

<u>7.2.3 The report_unique Command</u>	401
--	-----

<u>7.3 Coverage Grading in IMC</u>	407
--	-----

<u>7.3.1 Local Grade Calculation</u>	409
--	-----

<u>7.3.2 Cumulative Grade Calculation</u>	416
---	-----

A

<u>List of Attributes</u>	419
---------------------------------	-----

<u>A.1 Attributes Associated with Blocks</u>	419
--	-----

<u>A.2 Attributes Associated with Expressions</u>	422
---	-----

<u>A.3 Attributes Associated with Toggle Variables</u>	425
--	-----

<u>A.4 Attributes Associated with Toggle Signals</u>	427
--	-----

<u>A.5 Attributes Associated with States</u>	430
--	-----

<u>A.6 Attributes Associated with Transitions</u>	432
---	-----

<u>A.7 Attributes Associated with Assertions</u>	435
--	-----

<u>A.8 Attributes Associated with Cover Bins</u>	439
--	-----

<u>A.9 Attributes Associated with Instances and Types</u>	441
---	-----

B

<u>Known Gaps and Differences</u>	447
<u>B.1 Known Gaps in IMC in Comparison with ICCR</u>	447
<u>B.2 Known Gaps in IMC in Comparison with the Specman Coverage GUI</u>	447
<u>B.3 Differences from ICCR Functionality</u>	448
<u>Index</u>	1

Preface

This manual describes the Cadence® *Incisive Metrics Center* (IMC). IMC is an analysis tool available to display textual reports; and merge, rank, and refine metrics data.

Metrics data in verification is typically the coverage data coming from the simulator, formal tool, or the emulation environment.

This preface discusses the following topics:

- [Audience](#) on page 7
- [Related Documents](#) on page 7
- [Typographical Conventions](#) on page 8
- [Syntax Conventions](#) on page 8
- [About Online Help](#) on page 9
- [Customer Support](#) on page 9

Audience

This guide is intended for engineers who need to analyze metrics data from different simulation runs. Depending on your design and verification requirements, you should be familiar with Verilog HDL, VHDL, SystemC, SystemVerilog, and **e** language, and be familiar with the use of the Cadence Incisive or Palladium tools.

Related Documents

- For information on generating metrics data using Incisive Comprehensive Coverage, see *ICC User Guide*.
- For information on generating metrics data using Specman, see *Usage and Concepts Guide for e Testbenches*.
- For information on generating metrics data using Enterprise Manager, see *Incisive Enterprise Manager Getting Started*.

Typographical Conventions

The table lists the typographical conventions used in this manual.

Table 2-1 Typographical Conventions

Font	Meaning	Example
<i>italic</i>	Titles of books	See the <i>Incisive Enterprise Manager Getting Started</i> for more information.
literal	Program output or text that you type at the command line.	% imc -batch
<i>italic literal</i>	User-defined <i>arguments</i> for which you must substitute a name or a value.	% imc -load <run>

Syntax Conventions

The table lists the syntax conventions used in this manual.

Table 2-2 Syntax Conventions

Symbol	Meaning	Example
	Vertical bars (OR-bars) separate possible choices for a single argument. They take precedence over any other operator.	command <i>argument1</i> <i>argument2</i>
[]	Brackets denote optional arguments. When used with OR-bars, they enclose a list of choices. You can choose one argument from the list.	command [<i>argument1</i> <i>argument2</i>]
{ }	Braces are used with OR-bars and enclose a list of choices. You must choose one argument from the list.	command { <i>argument1</i> <i>argument2</i> }
...	Three dots (...) indicate that you can repeat the previous argument. If they are used within brackets, you can specify zero or more arguments. If they are used without brackets, you must specify at least one argument, but you can specify more.	<i>argument...</i> specify at least one [<i>argument</i>]... you can specify zero or more

About Online Help

The online documentation system is called Cadence Help.

Launching Cadence Help

- Set the path variable so that it includes the path to the executables, which are in `install_dir/tools/bin` and `install_dir/bin`, and then enter the following command at the prompt:
`cdnshelp &`
- You can also launch Cadence Help by selecting an item on the GUI *Help* menu or by clicking the *Help* button on forms and dialog boxes.

Getting Help for Cadence Help

After launching Cadence Help, press F1 or choose *Help - Contents* to display the help page for Cadence Help.

Customer Support

There are several ways that you can get help with your Cadence product:

- Customer support
Cadence is committed to keeping your design teams productive by providing answers to technical questions, the latest software updates, and education services to keep your skills updated. For information on Cadence support, go to the following web site:
<http://www.cadence.com/support>
- Cadence Online Support
Customers with a maintenance contract with Cadence can obtain current information on the tools at the following web site:
<http://support.cadence.com>
- Feedback about documentation
Contact Cadence Customer Support to file a CCR if you find:
 - An error in a manual
 - An omission of information in a manual

Incisive Metrics Center User Guide

Preface

- A problem using the Cadence Help documentation system

Getting Started with Incisive Metrics Center

Incisive Metrics Center (IMC) is an analysis tool available to merge metrics data, rank metrics data, display textual reports, refine metrics items to be excluded from coverage grade calculation, and analyze metrics data.

This chapter includes the following topics:

- [About IMC](#)
- [Quick Start Task Flow](#)
- [Launching IMC](#)
- [IMC Interface](#)
- [Loading Runs](#)
- [Metrics Page](#)
- [Managing Table Columns](#)
- [Filtering Table Data](#)
- [Defining and Organizing Views](#)
- [Exporting Metrics Tree to a CSV File](#)
- [Configuring Configurable Items](#)

1.1 About IMC

IMC is a metrics analysis tool that provides you an interactive way to evaluate:

- Code Coverage

Code coverage consists of block, expression, and toggle coverage. It measures how thoroughly a testbench exercises the lines of HDL code that describe a design. The coverage test results reveal areas of the design that have not been fully tested, or that did not meet a desired coverage criteria. When provided with this information, you can develop tests to target the untested or under-tested areas. For example, block coverage information can tell you whether or not various sections of the design were ever stimulated by test vectors. IMC shows this information in a visual and easily understandable way.

Note: Block coverage can be generated from Incisive or Specman. Using IMC, you can analyze, merge, rank, and report coverage generated from Incisive as well as Specman.

- **FSM Coverage**

FSM coverage interprets the synthesis semantics of the HDL design and monitors the coverage of the FSM representation of control logic blocks in the design. With FSM coverage, you can determine whether or not all possible states and transitions in a state machine were visited.

- **Functional Coverage**

Functional coverage is generated by inserting PSL, SystemVerilog assertions, or SystemVerilog covergroup statements into the code and simulating the design. The assertions create functional coverage points that indicate test scenarios you want to cover. IMC displays the coverage points and the number of times each one was covered. IMC also supports analysis, merging, and ranking of functional coverage defined in **e** language and collected by Specman.



You must add <install-dir>/bin to the environment path to launch and analyze coverage with IMC.

1.2 Quick Start Task Flow

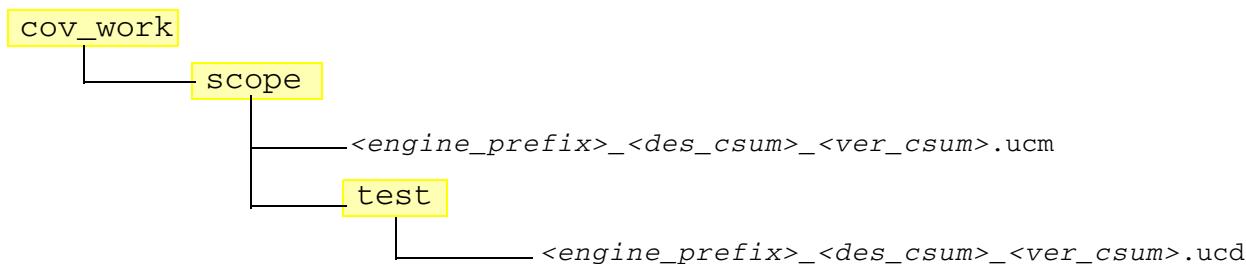
This section quickly describes the common task flow for using IMC.

1. Generate metrics data. Metrics data can be generated through Incisive or Specman. For more details, see *ICC User Guide* and *Usage and Concepts Guide for e Testbenches*.
2. Launch IMC. See [Launching IMC](#) for more details.
3. Load the metrics data into IMC. See [Loading Runs](#) for more details.

4. Start analyzing the metrics data. Start with the [Metrics Page](#) and identify interesting or problematic elements for detailed analysis.
5. Perform detailed analysis using individual analysis pages. See [Analyzing Metrics Data](#) for more details.

Metrics Data Storage

IMC reads metrics data from the run directories. A run directory contains all coverage databases from Specman and Incisive that are relevant to a single run. By default, metrics data is stored as:



where

- `<engine_prefix>` is the name of the engine using which coverage data is generated. It is `sn` for Specman, `icc` for Incisive, and `ifv` for Incisive Formal Verifier (IFV).
- `<des_csum>` is an 8-digit hexadecimal number which is a unique-identifier (checksum) of design hierarchy and code coverage metrics, such as block, expression, and so on.
- `<ver_csum>` is an 8-digit hexadecimal number which is a unique-identifier (checksum) of design hierarchy and verification (functional) coverage metrics namely, assertion and covergroup.

Note: The `test` directory may also include `icc.com` file if COM is enabled, and a `.vsوف` file if vsوف dumping is enabled.

For more details on coverage data generation, see *ICC User Guide* and *Usage and Concepts Guide for e Testbenches*.

1.3 Launching IMC

You can launch IMC in the following modes:

- GUI mode

To launch IMC in GUI mode, type `imc` at the command prompt and press **Enter**. This shows the splash screen, as shown in [Figure 1-1](#) on page 14 and then displays the *Incisive Metrics Center* window.

- **Interactive Command-line mode**

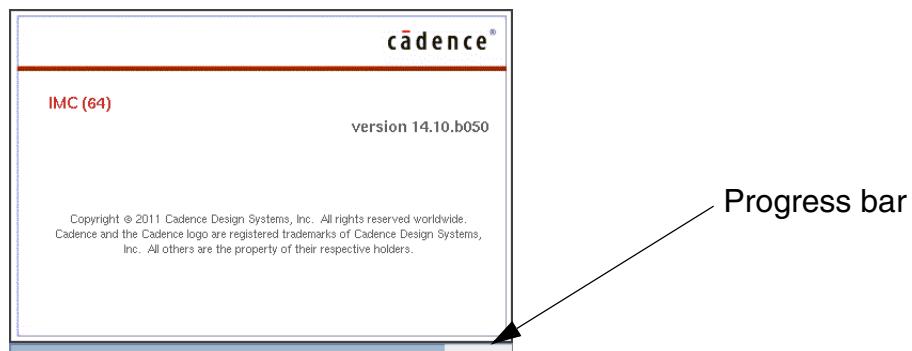
To launch IMC in interactive mode, type `imc -batch` at the command prompt and press **Enter**. This brings up the `imc` command prompt and you can issue IMC commands. In this mode, control returns to you after execution of each command. For more details, see [IMC Command-Line Interface](#) on page 237.

- **Batch mode**

To launch IMC in the batch mode, type `imc -exec <command_file>` at the command prompt and press **Enter**. In this mode, all commands are placed in a `<command_file>` which is passed to IMC for execution. When you run the `imc -exec <command_file>` command, IMC is launched, all commands in the file `<command_file>` are executed, and after execution of all of the commands, IMC exits. If any of the commands in this file fails, IMC terminates without executing the remaining commands. For more details, see [IMC Command-Line Interface](#) on page 237.

[Figure 1-1](#) on page 14 displays the splash screen that appears at the time of launching IMC.

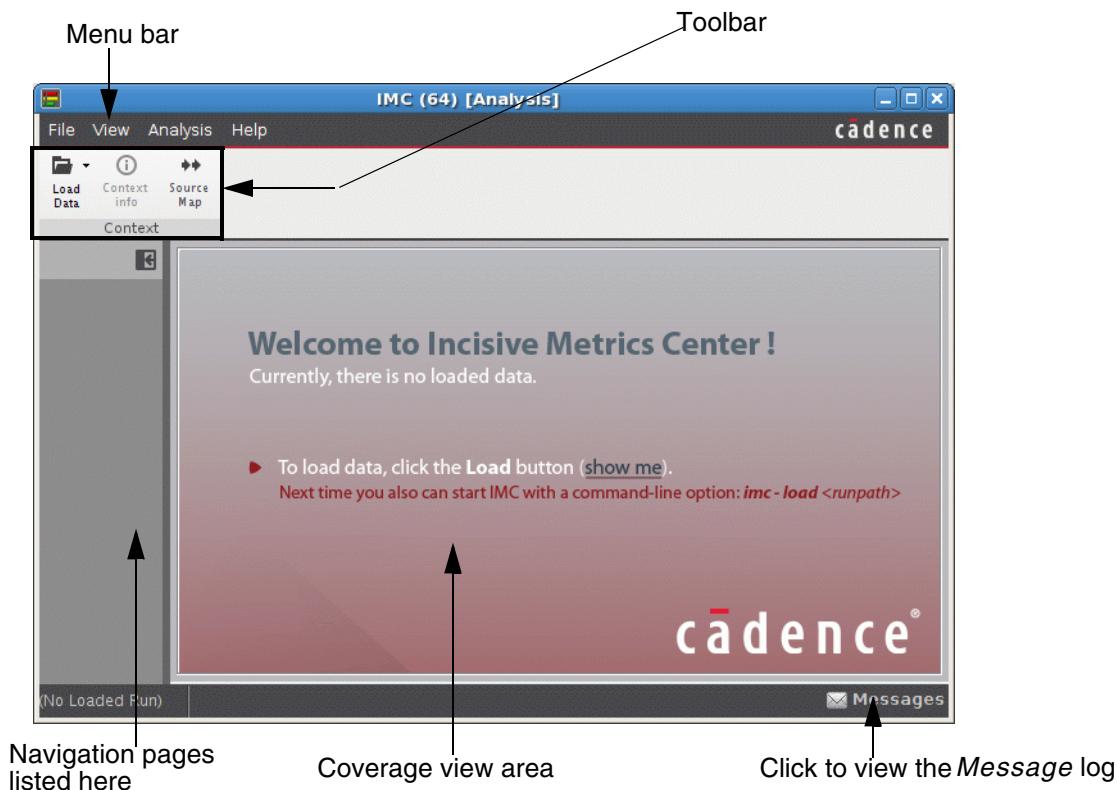
Figure 1-1 IMC Splash Screen



The progress bar at the bottom of the splash screen indicates how much time remains until the application is fully loaded.

After the splash screen, the *Incisive Metrics Center* window is displayed, as shown in [Figure 1-2](#) on page 15.

Figure 1-2 IMC Window



1.3.1 Launching IMC in 32-Bit Mode

By default, 64-bit version of IMC is launched. You can launch IMC in 32-bit mode in any of the following ways:

- Using the `-32bit` command-line option as:
`imc -32bit`
- Setting any of the following environment variables:
 - `setenv IMC_AUTO_32BIT ALL`

The advantage of setting the environment variable is that you do not have to include the `-32bit` option on the command line every time when you have to launch the tool.

Note: If you set this environment variable and also include the `-64bit` option on the command line, then tool is launched in 64-bit mode.

If you set both `CDS_AUTO_64BIT ALL` and `IMC_AUTO_32_BIT ALL`, then 64-bit version of the tool will be launched because `CDS_AUTO_64BIT` is stronger than the `IMC_AUTO_32_BIT`.

Note: To confirm which version is currently set up, use the `ncbits` command. This command returns either 32 or 64.

```
% ncbits  
64
```

1.3.2 Launching IMC from Enterprise Manager

You can also launch IMC from Enterprise Manager. For details on launching IMC from Enterprise Manager, see *Invoking IMC* in the *Incisive Enterprise Manager Analyzing Metrics* guide.

1.3.3 Launching IMC from SimVision

You can also launch IMC from SimVision. For details on launching IMC from SimVision, see the *Managing Coverage for e Testbenches* guide.

1.4 IMC Interface

The IMC window has the following components:

- Menu Bar
- Toolbar
- Navigation Pages
- Coverage View Area
- Messages Log

Note: Users that use KDE desktop, might have problems when minimizing a dialog. The application might become unresponsive and the dialog gets hidden. For more details, see: https://bugs.kde.org/show_bug.cgi?id=277847 or <http://permalink.gmane.org/gmane.comp.kde.devel.bugs/1498754>.

1.4.1 Menu Bar

The menu bar, shown in [Figure 1-3](#) on page 17, contains commands that you use to load runs, view coverage information in different ways, open separate analysis pages, and read documentation. The menu bar is common to all navigation pages.

Figure 1-3 Menu Bar



The menu bar has the following menus:

- [File](#)
- [View](#)
- [Analysis](#)
- [Help](#)

1.4.1.1 File

The *File* menu contains the commands for loading run, unloading run, and exiting IMC. You can choose the following items from the *File* menu:

- Load—To load run
- Reload—To reload the already loaded run
- Unload—To unload the loaded run
- Exit—To exit the tool

Note: The *File* menu also maintains a history of previously loaded runs (in current or previous IMC runs). These runs are also listed in the *File* menu, as shown in [Figure 1-4](#) on page 18. A maximum of nine runs are maintained and listed in the *File* menu.

Figure 1-4 File Menu



You can quickly load a run by selecting it.

1.4.1.2 View

The *View* menu contains the commands for enabling pop-ups, disabling pop-ups, organizing dialog choices, and removing dialog choices. You can choose the following items from the *View* menu:

- Toolbars
- Organize Message Popups
- Allow Showing All Message Popups
- Skip All Message Popups
- Organize Dialog Choices
- Allow Showing All Dialogs
- Save Settings
- Reset Settings
- Auto Save Settings
- Restore Current Layout
- Enable Anti-Aliasing
- Configuration

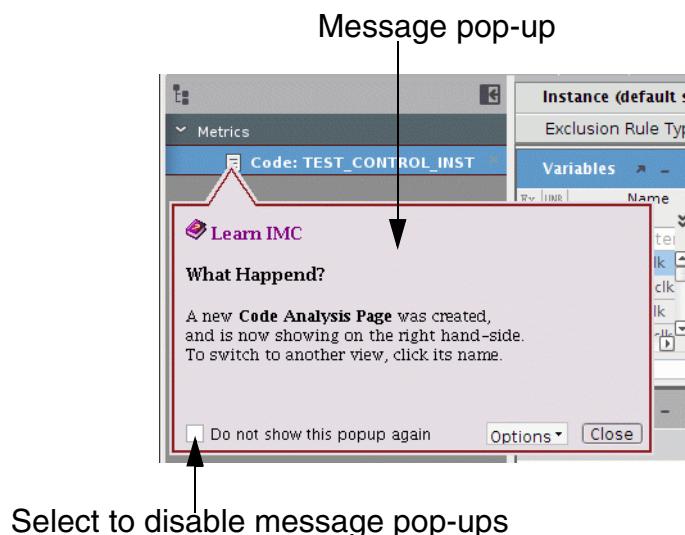
Toolbars

The *Toolbars* option of the *View* menu displays a list of toolbars that can be shown or removed from the current IMC window. Using this option, you can enable or disable the showing of a specific toolbar from the IMC window.

Organize Message Popups

By default, whenever you perform an action such as launching a new analysis page or selecting a specific view, a message pop-up window, as shown in [Figure 1-5](#) on page 19 is displayed. This pop-up describes what happened because of the action performed.

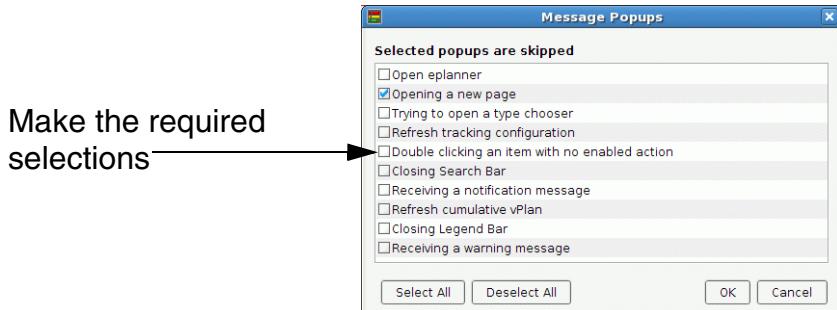
Figure 1-5 Pop-Up Window



You can disable such pop-ups by selecting the check box available on the message pop-up box.

The *Organize Message Popups* option of the *View* menu allows you to specify the scenarios for which message pop-ups must be displayed. [Figure 1-6](#) on page 20 shows the *Message Popups* dialog box that is displayed when you select the *Organize Message Popups* option from the *View* menu.

Figure 1-6 Pop-Up Window



In the *Message Popups* dialog box, you can make the required selections or deselections. The message pop-up is disabled/skipped for the items you select in this dialog box.

Allow Showing All Message Popups

The *Allow Showing All Message Popups* option of the *View* menu enables showing all message pop-ups. It is the same as deselecting all of the items on the *Message Popups* dialog box.

Skip All Message Popups

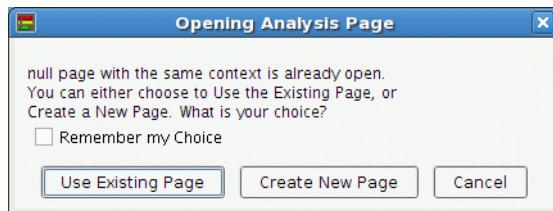
The *Skip All Message Popups* option of the *View* menu disables showing all message pop-ups. It is the same as selecting all of the items on the *Message Popups* dialog box.

Organize Dialog Choices

In the IMC GUI, when you perform certain actions, such as, opening an already-launched analysis page, or opening an analysis page for items that have cumulative data but no local data, a dialog box appears to indicate what happened.

Consider an example, where you already have a code coverage page launched for an entity *RAM_256x16_TEST_INST*. Later, if you try to launch a code coverage page again for the same entity, a dialog box, as shown in [Figure 1-7](#) on page 21 will be displayed.

Figure 1-7 Opening Analysis Page



This dialog box indicates that the page is already launched. It also prompts you to take an appropriate action, such as *Use Existing Page* or *Create New Page*.

This dialog box also enables remembering the choice you made in that scenario. If you select the check box, then the choice made on that dialog box is preserved for reuse in similar scenarios.

If you select the *Remember my Choice* check box and select *Use Existing Page*:

- A new page will not be launched and the already launched page will be used for further analysis, and
- Your action that is *Use Existing Page* will be preserved for reuse in similar scenarios.

Note: Next time, if you try to launch an already-launched page, you will not be prompted with this dialog box. The tool will automatically use the existing launched page because you selected the *Remember my Choice* check box. The choice made is reused if the same scenario is repeated, either in the same invocation of IMC, or in a different invocation.

The choices that you preserve in different dialog boxes can be selected or deselected using the *Organize Dialog Choices* option of the *View* menu.

When you select the *Organize Dialog Choices* option from the *View* menu, the *Dialog Choices* dialog box, as shown in [Figure 1-8](#) on page 21 is displayed.

Figure 1-8 Dialog Choices



The *Dialog Choices* dialog box displays a list of choices saved on different dialog boxes. This list keeps appending as you save more choices. The dialog boxes listed here are

skipped, and the choice saved for the listed dialog boxes are used. You can remove an already saved choice by deselecting it from the list and clicking *OK*.

Allow Showing All Dialogs

The *Allow Showing All Dialogs* option removes all of the saved choices. It is the same as deselecting all of the choices on the *Dialog Choices* dialog box.

Save Settings

The *Save Settings* option allows you to preserve the settings, such as the size of the IMC window and size of tables, for reuse in the same as well as the next run of IMC.

Reset Settings

The *Reset Settings* option allows you to reset all of the settings and enable the default settings.

Note: This option does not enable the default settings immediately. Instead, the default settings take effect only on the next invocation of IMC.

This option hides the *Lookup* toolbar wherever applicable. For example, if the *Lookup* toolbar is shown on the IMC screen and you select the *Reset Settings* option from the *View* menu, then next time when you invoke IMC, the *Lookup* toolbar will be hidden (not shown on the IMC screen). Later, when required, you can launch the *Lookup* toolbar from the *View* menu.

Auto Save Settings

By default, IMC saves the settings, such as the size of the IMC window, size of tables that you set during the IMC run. These settings are automatically saved and reapplied to the next IMC run.

To disable this automatic saving of settings, deselect the *Auto Save Settings* check box in the *View* menu.

Restore Current Layout

While working in IMC, you might change the settings of view area, such as detach a pane, resize a pane, or hide a pane. The *Restore Current Layout* option helps you restore the original settings of the IMC view area.

For more details on detaching a pane, see [Toggle Floating](#) and for more details on hiding a pane, see [Hide](#).

Enable Anti-Aliasing

Anti-aliasing is a process where the edges of images and fonts have pixels added to blend the edge of the item into the background. The goal of anti-aliasing is to make the fonts and images appear smoother on a computer monitor.

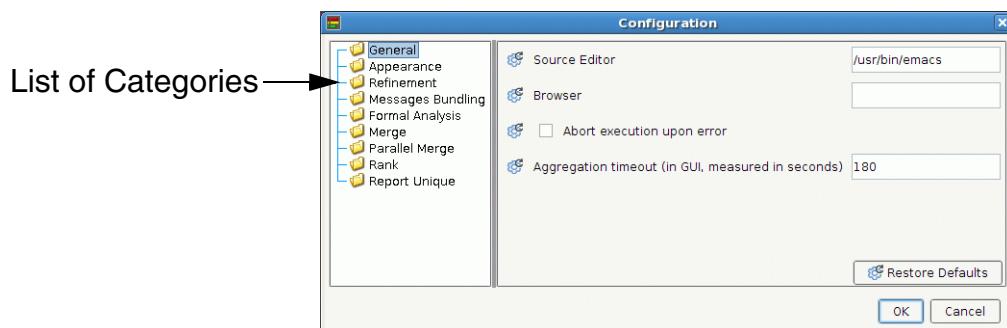
By default, this option is enabled. You can turn OFF anti-aliasing by clearing this check box.

Configuration

The *Configuration* option displays a dialog box using which you can configure options related to appearance, refinement, and message logs, and so on.

[Figure 1-9](#) on page 23 displays the *Configuration* dialog box.

Figure 1-9 Configuration



For more details, see [Configuring Configurable Items](#) on page 78.

1.4.1.3 Analysis

The *Analysis* menu has following options:

- Load Data—To load a run. For more details, see [Loading Runs](#) on page 39.
- Context information—To view the name and location of the loaded run, the UCD file, the UCM file, the coverage configuration file (CCF), and the loaded refinement file. For more details, see [Context](#) on page 26.
- Source Map —To map the source to new paths. For more details, see [Source Map](#).

- Block Analysis—To launch a separate page for more detailed block analysis of selected instance or type. For more details, see [Chapter 2, “Analyzing Metrics Data.”](#)
- Expression Analysis—To launch a separate page for more detailed expression analysis of selected instance or type. For more details, see [Chapter 2, “Analyzing Metrics Data.”](#)
- Toggle Analysis—To launch a separate page for more detailed signal analysis of selected instance or type. For more details, see [Chapter 2, “Analyzing Metrics Data.”](#)
- FSM Analysis—To launch a separate page for more detailed FSM analysis of selected instance or type. For more details, see [Chapter 2, “Analyzing Metrics Data.”](#)
- Cover Group Analysis—To launch a separate page for more detailed cover group analysis of selected instance or type. For more details, see [Chapter 2, “Analyzing Metrics Data.”](#)
- Assertion Analysis —To launch a separate page for more detailed assertion analysis of selected instance or type. For more details, see [Chapter 2, “Analyzing Metrics Data.”](#)
- Lookup Type—To search for the type of the selected instance. For more details, see [Lookup on page 31](#).
- Lookup Entity in Hierarchy—To lookup for the selected instance or type in the verification hierarchy pane. For more details, see [Lookup on page 31](#).
- Lookup Parent Entity in Hierarchy—To lookup for the parent of the selected instance or type in the verification hierarchy pane. For more details, see [Lookup on page 31](#).
- Exclude —To exclude the selected item and its children. For more details, see [Chapter 3, “Refining Metrics Data.”](#)
- Exclude Local Aspects only—To exclude only top-level instances. For more details, see [Chapter 3, “Refining Metrics Data.”](#)
- Exclude Smart — To exclude such that any entity connected to the excluded entity will be excluded implicitly. For more details, see [Chapter 3, “Refining Metrics Data.”](#)
- Show Affected — To show the list of entities that were implicitly marked excluded because of the *Exclude Smart* action on that entity. For more details, see [Chapter 3, “Refining Metrics Data.”](#)
- Show Affecting — To show the list of entities that caused the indirect smart exclusion. For more details, see [Chapter 3, “Refining Metrics Data.”](#)
- Edit Comment —To edit exclusion comment. For more details, see [Chapter 3, “Refining Metrics Data.”](#)
- Un-Exclude—To un-exclude already excluded items. For more details, see [Chapter 3, “Refining Metrics Data.”](#)

- Clear dirty rules —To clear/delete the dirty rules. This option is available recursively from the type/instance/fsm parent entity. It is also available from the dirty excluded entity itself. For more details, see [Chapter 3, “Refining Metrics Data.”](#)
- Approve dirty rules — To approve the dirty rules (make the rule completely valid). This option is available recursively from the type/instance/fsm parent entity. It is also available from the dirty excluded entity itself. For more details, see [Chapter 3, “Refining Metrics Data.”](#)
- Clear all orphan rules—To clear/delete all the orphan rules of the selected item. This option is available recursively from the type/instance/fsm parent entity. For more details, see [Chapter 3, “Refining Metrics Data.”](#)
- Apply all orphan rules —To apply all the orphan rules of the selected item (by index of originally excluded entity) and clear them. This option is available recursively from the type/instance/fsm parent entity. For more details, see [Chapter 3, “Refining Metrics Data.”](#)
- Read Refine—To load already saved exclusion file. For more details, see [Chapter 3, “Refining Metrics Data.”](#)
- Save Refine—To save the exclusions to a file for later use. For more details, see [Chapter 3, “Refining Metrics Data.”](#)
- Reports —To generate metrics reports. For more details, see [Chapter 7, “Reporting and Grading.”](#)

1.4.1.4 Help

The *Help* menu provides you with access to the online help. You can choose:

- *IMC User Guide*—To display the *Incisive Metrics Center User Guide*
- *IMC Quick Reference*—To display the *Incisive Metrics Center Quick Reference Guide*
- *IMC KPNS*—To display the *Incisive Metrics Center Known Problems and Solutions* document
- *IMC What’s New*—To display the *Incisive Metrics Center What’s New* document
- *Cadence Help Library*—To launch *Cadence Help* with which you can view the currently installed documents
- *Cadence Online Support*—To open an Internet browser and launch the www.support.cadence.com page
- *About IMC* —To display the version and copyright information of the IMC GUI

Note: IMC documentation can also be launched from the command-line mode. For more details, see [Launching Documentation](#) on page 256.

1.4.2 Toolbar

The following toolbars are available on the *Metrics* page:

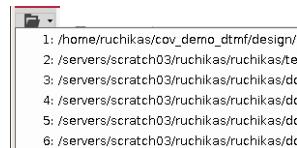
- [Context](#)
- [Views](#)
- [Analyze](#)
- [Lookup](#)
- [Refinement](#)
- [Report](#)

1.4.2.1 Context

The *Context* toolbar has following items:

- A *Load* button—It allows you to navigate through the file hierarchy, and select a database for loading. See [Loading Runs](#) on page 39 for more details on loading runs.
- A drop-down next to the *Load* button, as shown in [Figure 1-10](#) on page 26 — It displays a list of previously loaded runs (in current or previous IMC runs). A maximum of nine runs are maintained and listed in the drop-down next to the *Load* button.

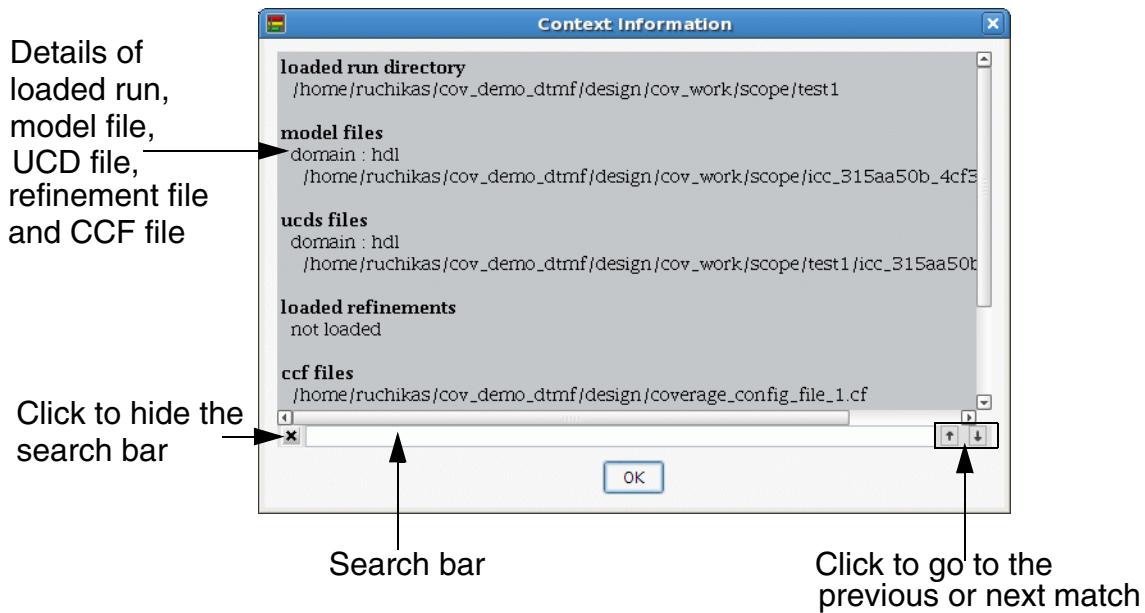
Figure 1-10 Drop-Down next to Load Button



This drop-down allows you to load a previously loaded database through a simple selection.

- A *Context info* button — It allows you to view the name and location of the loaded run, the UCM file, the UCD file, the loaded refinement files, and the location and contents of the coverage configuration file (CCF). When you click the *Context info* button, the *Context Information* dialog box is displayed, as shown in [Figure 1-11](#) on page 27.

Figure 1-11 Context Information Dialog Box

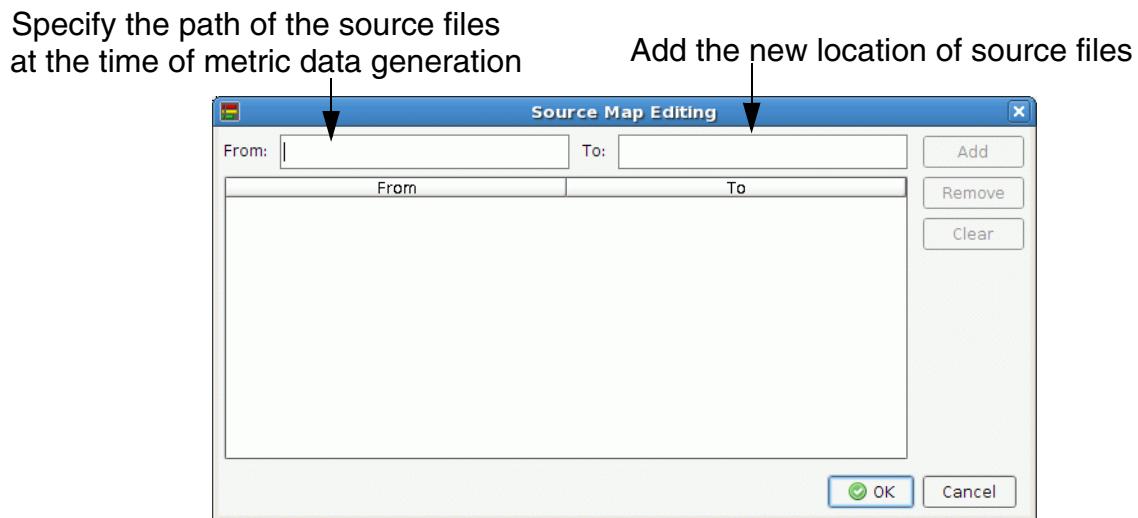


In addition to viewing the context information, you can perform a search on the information displayed in the *Context Information* dialog box.

To quickly search for a specific text, type the search text in the text box. As you type the text in the text box, the text gets highlighted in the context information. You can use the *Up* and *Down* arrow keys to go to previous or next match. You can hide the search bar by clicking the *X* icon shown on the search bar. To reopen the search bar, press *Ctrl + F* keys.

- A Source Map button — It allows you to map the source to new paths or remove existing mapped sources. This feature is useful if source files are moved, or are referenced differently on different systems. For example, if the source files are moved to a different location, then during data analysis the *Source* tab page does not show the source code. To resolve the issue and to show the source code, you can map the source path to a new location where the source files are located. For this, click the *Source Map* button in the *Context* toolbar. When you click the *Source Map* button, the *Source Map Editing* dialog box is displayed, as shown in [Figure 1-12](#) on page 28.

Figure 1-12 Source Map Editing Dialog Box



- ❑ In the *From* text box, specify the path of the source files at the time of metrics data generation.
- ❑ In the *To* text box, specify the new location of the source files.

For example, if the original source location was `/home/usr1/temp/src` and the new location of the source files is `/home/usr1/src_files/vlog`, then specify `/home/usr1/temp/src` in the *From* text box and specify `/home/usr1/src_files/vlog` in the *To* text box.

- ❑ Click *Add* to map the old path (*From*) to the new path (*To*) and click *OK*.

Note: If the suffix of the source path is same, then instead of specifying the complete path in the *From* and *To* fields, you can consider specifying just the prefix of the path. For example, if the original source location is `/home/usr1/temp/src` and the new location of the source files is `/home/usr2/temp/src`, then you can specify `/home/usr1` in the *From* text box and `/home/usr2` in the *To* text box.

After adding a source map path, you must reload the run to ensure that the *Source* tab page shows the source code based on the path you mapped in the *Source Map Editing* dialog box.

Note: The *Remove* button allows you to remove an already mapped path. You can select an already mapped path and click *Remove* to delete the path. The *Clear* button allows you to clear all the mapped paths together. For example, to clear all the mapped paths, click the *Clear* button. It will remove all the mapped paths together.

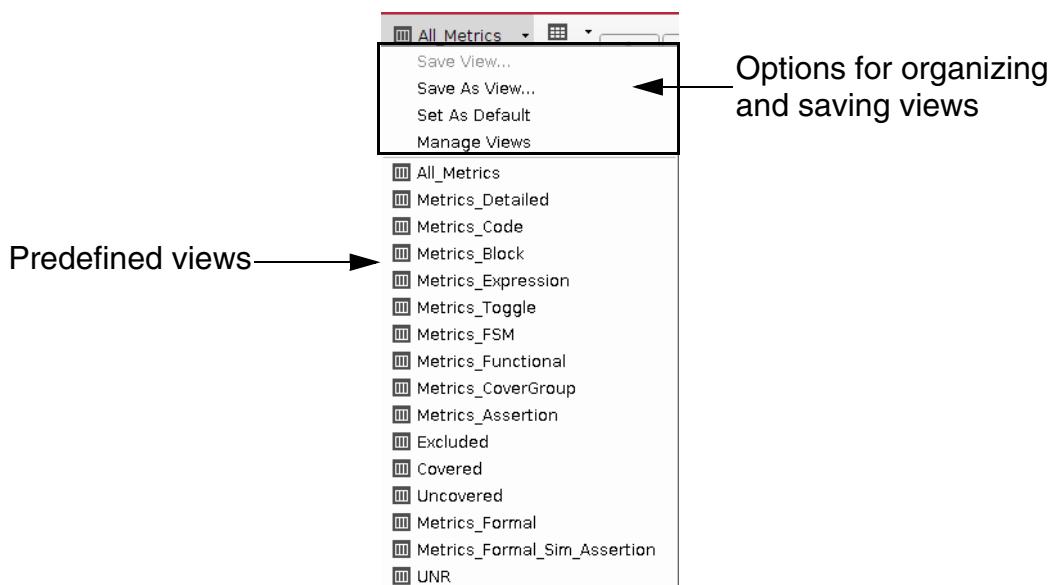
Mapping of source paths is also available in command-line mode of IMC. For details on mapping source paths in command-line mode, see [Mapping Source Path](#) on page 247.

1.4.2.2 Views

The *Views* toolbar has a drop-down button, which shows the currently selected view. By default, when you load a run, *All_Metrics* view is selected. When you click the drop-down button, it shows the options using which you can define views, organize views, set views, or quickly launch a specific predefined view.

[Figure 1-13 on page 29](#) shows the options available in the *All_Metrics* drop-down list.

Figure 1-13 All Metrics View



You can select any of the following predefined views:

- *All_Metrics*—To show *Overall Average Grade*, *Overall Covered*, and *Assertion Status Grade* attributes in the *Verification Hierarchy* pane.
- *Metrics_Detailed*—To show *Overall Average Grade*, *Overall Covered*, *Block Average Grade*, *Block Covered*, *Expression Average Grade*, *Expression Covered*, *Toggle Average Grade*, *Toggle Covered*, *State Average Grade*, *State Covered*, *Transition Average Grade*, *Transition Covered*, *Assertion Average Grade*, *Assertion Covered*, *Covergroup Average Grade*, and *Covergroup Covered* attributes in the *Verification Hierarchy* pane.
- *Metrics_Code*—To show only *Code Average Grade*, *Code Covered*, and *Valid Metrics* attributes in the *Verification Hierarchy* pane.
- *Metrics_Block*—To show only *Block Average Grade*, *Block Covered*, and *Valid Metrics* attributes in the *Verification Hierarchy* pane.

- Metrics_Expression—To show only *Expression Average Grade*, *Expression Covered*, and *Valid Metrics* attributes in the *Verification Hierarchy* pane.
- Metrics_Toggle—To show only *Toggle Average Grade*, *Toggle Covered* and *Valid Metrics* attributes in the *Verification Hierarchy* pane.
- Metrics_FSM—To show only *FSM Average Grade*, *FSM Covered*, and *Valid Metrics* attributes in the *Verification Hierarchy* pane.
- Metrics_Functional—To show only *Functional Average Grade*, *Functional Covered* and *Valid Metrics* attributes in the *Verification Hierarchy* pane.
- Metrics_Covergroup—To show only *Covergroup Average Grade*, *Covergroup Covered* and *Valid Metrics* attributes in the *Verification Hierarchy* pane.
- Metrics_Assertion—To show *Assertion Average Grade*, *Assertion Covered*, *Assertion Status Grade*, and *Valid Metrics* attributes in the *Verification Hierarchy* pane.
- Excluded—To show *Overall Average Grade*, *Overall Covered*, *Assertion Status Grade*, and *Overall Local Excluded* attributes in the *Verification Hierarchy* pane. When you select this view, the table data is filtered to show items with *Overall Local Excluded* > 0. For more details on filtering values in verification hierarchy tree, see [Quick Filter Bar](#) on page 55.
- Covered—To show *Overall Average Grade*, *Overall Covered*, *Overall Local Grade*, and *Overall Local Covered* attributes in the *Verification Hierarchy* pane. When you select this view, the table data is filtered to show items with *Overall Local Grade* == 100. For more details on filtering values in verification hierarchy tree, see [Quick Filter Bar](#) on page 55.
- Uncovered—To show *Overall Average Grade*, *Overall Covered*, *Overall Local Grade*, and *Overall Local Covered* attributes in the *Verification Hierarchy* pane. When you select this view, the table data is filtered to show items with *Overall Local Grade* in the range of 0..0 and 99..99. For more details on filtering values in verification hierarchy tree, see [Quick Filter Bar](#) on page 55.
- Metrics_Formal—To show *Formal Average Grade*, *Formal Covered*, *Formal Status Grade*, and *Valid Metrics* attributes in the *Verification Hierarchy* pane.
- Metric_Formal_Sim_Assertion—To show *Assertion Average Grade*, *Assertion Covered*, *Assertion Status Grade*, *Formal Average Grade*, *Formal Covered*, *Formal Status Grade*, and *Valid Metrics* attributes in the *Verification Hierarchy* pane.
- UNR—To show only the items that are marked unreachable. This view shows only *Overall Local UNR* and *Overall UNR* attributes in the *Verification Hierarchy* pane.

Note: *Valid Metrics* attribute has a pre-defined filter already set based on the view you select. For example, if you select *Metrics_Code*, then the filter field will show a value [b, e, t] and will filter the verification hierarchy to show the types and instances in which blocks, expressions, and toggles are scored. For more details on filtering values in verification hierarchy tree, see [Quick Filter Bar](#) on page 55.

Note: Apart from the above predefined views, you can set and organize views. For more details, see [Defining and Organizing Views](#) on page 72.

1.4.2.3 Analyze

The *Analyze* toolbar, shown in [Figure 1-14](#) on page 31 allows you to launch a separate page for more detailed analysis of the selected item.

Figure 1-14 Analyze



For example, to perform detailed block coverage analysis of an instance, select the instance and then select *Block* from the *Analyze* toolbar. This will launch a separate page for block analysis.

Note: You can also select the above options from the *Analysis* menu.

Note: In the *Analyze* toolbar, only the analysis applicable to the selected instance/type show as enabled.

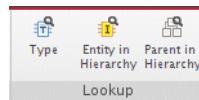
See [Analyzing Metrics Data](#) on page 105 for more information.

1.4.2.4 Lookup

The *Lookup* toolbar, shown in [Figure 1-15](#) on page 32 allows you to search for and select the type of a selected instance, look for the instance or type in the hierarchy tree, or look for the parent instance of the selected instance.

Note: Currently, IMC does not allow accessing instances if the number of child instances under a single parent instance (listed in the *Verification Hierarchy* pane) exceeds the predefined limit of 2500.

Figure 1-15 Lookup



The *Lookup* toolbar has the following options:

- [Lookup Type](#)
- [Lookup Entity in Hierarchy](#)
- [Lookup Parent in Hierarchy](#)

Note: You can also select the above options from the *Analysis* menu.

Lookup Type

The *Lookup Type* option is used to search and select the type of a selected instance in the *Verification Hierarchy* pane.

You can select the instance either in the *Instance* tree or the *Relative Elements* tab page of the *List tabs* pane.

Lookup Entity in Hierarchy

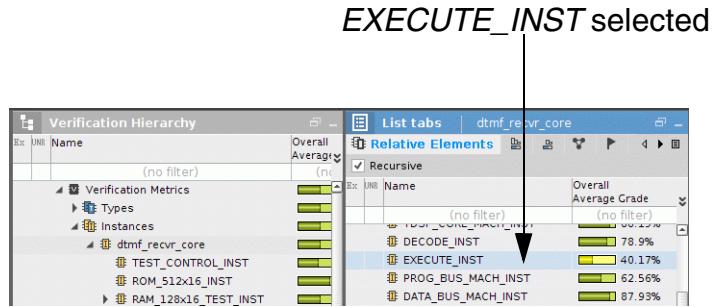
The *Lookup Entity in Hierarchy* option is used to look up the selected instance or type in the *Verification Hierarchy* pane.

This option is available only for the selections made in the *List tabs* pane. You can select the type or instance in the *Relative Elements* tab page of the *List tabs* pane and then select the *Lookup Entity in Hierarchy* option. This will immediately take you to the selected instance or type in the *Verification Hierarchy* pane.

For example, to look up for instance *EXECUTE_INST* in the *Verification Hierarchy* pane:

1. Select *EXECUTE_INST* in the *Relative Elements* tab page of the *List tabs* pane, as shown in [Figure 1-16](#) on page 33.

Figure 1-16 Lookup Entity in Hierarchy

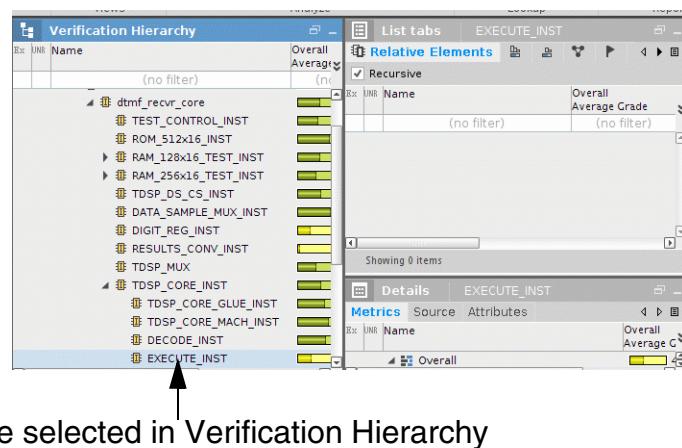


2. Select *Entity in Hierarchy* from the *Lookup* toolbar.

Note: Alternatively, you can right-click the instance in the *Relative Elements* tab page of the *List tabs* pane and select *Lookup Entity in Hierarchy* from the pop-up menu.

The selected instance gets highlighted in the *Verification Hierarchy* pane, as shown in [Figure 1-17](#) on page 33.

Figure 1-17 Instance Highlighted in Verification Hierarchy Pane



Lookup Parent in Hierarchy

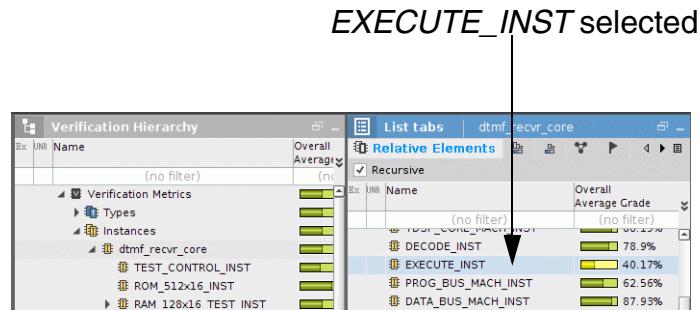
The *Lookup Parent Entity in Hierarchy* option is used to look up the parent of the selected instance or type in the *Verification Hierarchy* pane.

This option is available only for the selections made in the *List tabs* pane. You can select the type or instance in the *Relative Elements* tab page of the *List tabs* pane and then select the *Lookup Parent Entity in Hierarchy* option. This will immediately take you to the parent of the selected instance or type in the *Verification Hierarchy* pane.

For example, to look up for parent of instance *EXECUTE_INST* in the *Verification Hierarchy* pane:

1. Select *EXECUTE_INST* in the *Relative Elements* tab page of the *List tabs* pane, as shown in [Figure 1-18](#) on page 34.

Figure 1-18 Lookup Parent Entity in Hierarchy

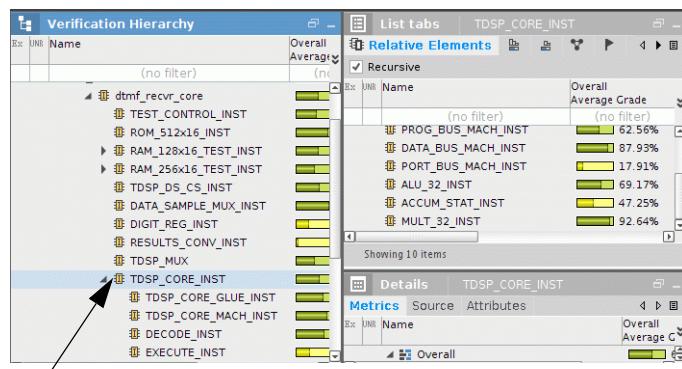


2. Select *Lookup Parent Entity in Hierarchy* from the *Lookup* toolbar.

Note: Alternatively, you can right-click the instance in the *Relative Elements* tab page of the *List tabs* pane and select *Lookup Parent Entity in Hierarchy* from the pop-up menu.

The selected instance gets highlighted in the *Verification Hierarchy* pane, as shown in [Figure 1-19](#) on page 34.

Figure 1-19 Parent Instance Highlighted in Verification Hierarchy Pane

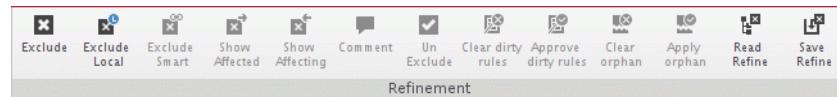


Parent instance selected in Verification Hierarchy

1.4.2.5 Refinement

The *Refinement* toolbar, shown in [Figure 1-20](#) on page 35 provides options that allow you to exclude items from coverage analysis, un-exclude excluded items, save refinements, and read refinements during coverage analysis.

Figure 1-20 Refinement Toolbar



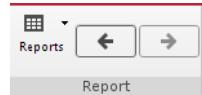
You can also select the above options from the *Analysis* menu.

For details on excluding metrics items, see [Refining Metrics Data](#) on page 167.

1.4.2.6 Report

[Figure 1-21](#) on page 35 shows the *Report* toolbar.

Figure 1-21 Report Toolbar



The *Report* toolbar has following options:

- [Report](#)
- [Selection History Buttons](#)

Note: You can also select the above options from the *Analysis* menu.

Report

The *Report* button in the *Report* toolbar allows you to generate metrics report from IMC GUI.

For more details, see [Generating Reports in GUI](#) on page 315.

Selection History Buttons

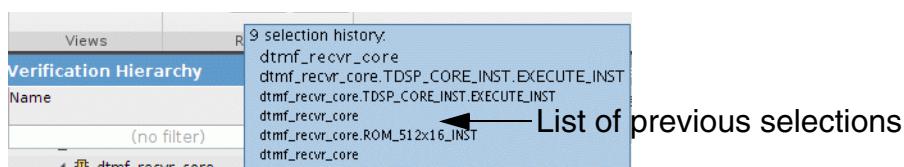
While navigating through the design, the selections that you make in the various panes of the *Metrics* page are stored and maintained by IMC until you reload the run. You can quickly navigate through those selections using the *Selection History* navigation buttons, as shown in [Figure 1-22](#) on page 36.

Figure 1-22 Selection History Navigation Buttons



When you place the cursor over the navigation buttons, a list of selections is displayed. [Figure 1-23](#) on page 36 displays a list of previous selections.

Figure 1-23 List of Selections

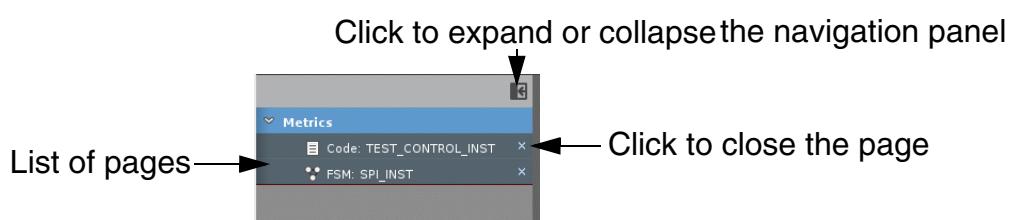


Placing the cursor over the next arrow key will display the selections made after the current selection. You can click the next arrow key to quickly reach the next selection.

1.4.3 Navigation Pages

Navigation pages are listed in the left side panel of the IMC window, as shown in [Figure 1-24](#) on page 36.

Figure 1-24 Navigation Pages



When you load a run, the only page listed in the *Navigation* panel is the *Metrics* page. Later, as you open more pages for detailed analysis, a different page is opened and is added to the

list of pages in the *Navigation* panel. You can switch from one page to another by clicking on the page in the *Navigation* panel. To close a page, click the *X* button next to the page name.

You can also right-click on a page to display a pop-up menu, which has the following items:

- Close —To close the page
- Close All Closable Pages—To close all of the pages that can be closed
- Rename—To [Rename a Page](#)

1.4.3.1 Rename a Page

To rename a page:

1. Right-click the page in the *Navigation* panel, and select *Rename*.

Note: Alternatively, double-click the context name in the *Navigation* pane.

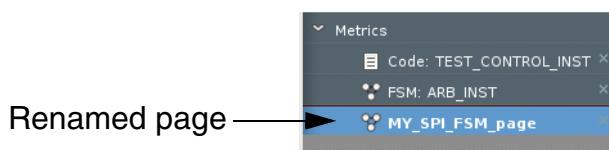
2. The cursor appears in the *Navigation* pane, as shown in [Figure 1-25](#) on page 37 and allows inline editing of the name of the page. Specify the new name for the page and press *Enter*.

Figure 1-25 Rename a Page



If you specify the name as `MY_SPI_FSM_page`, then the renamed page is shown in the *Navigation* pane, as shown in [Figure 1-26](#) on page 37.

Figure 1-26 Navigation Pane with Renamed Page

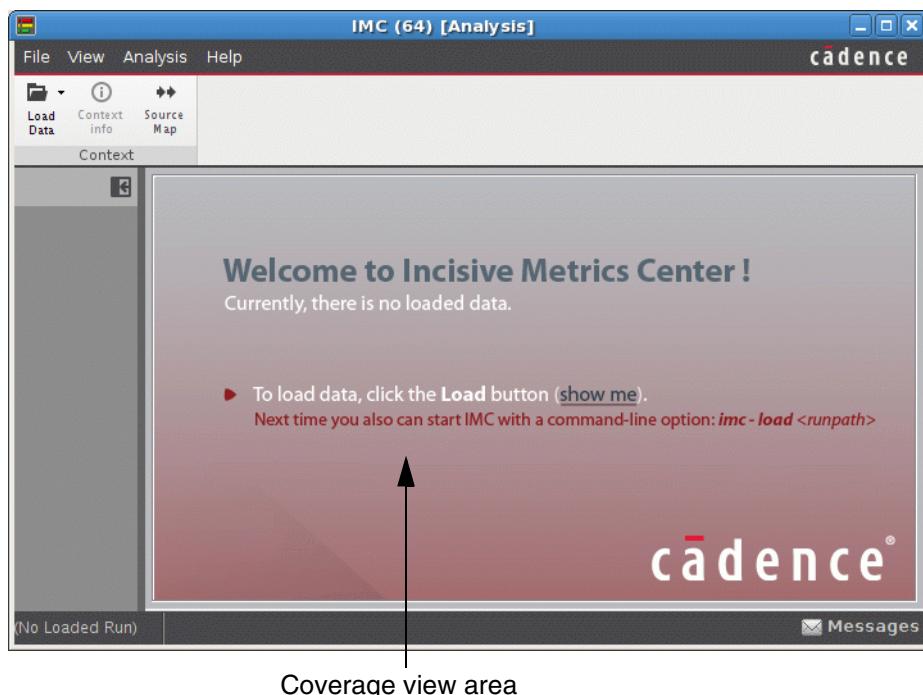


3. Notice that the `SPI_INST` page is renamed as `MY_SPI_FSM_page`.

1.4.4 Coverage View Area

When you launch IMC, the Coverage View area displays a welcome screen and instructions on how to proceed. The Coverage View area when you launch IMC is shown in [Figure 1-27](#) on page 38.

Figure 1-27 Coverage View Area

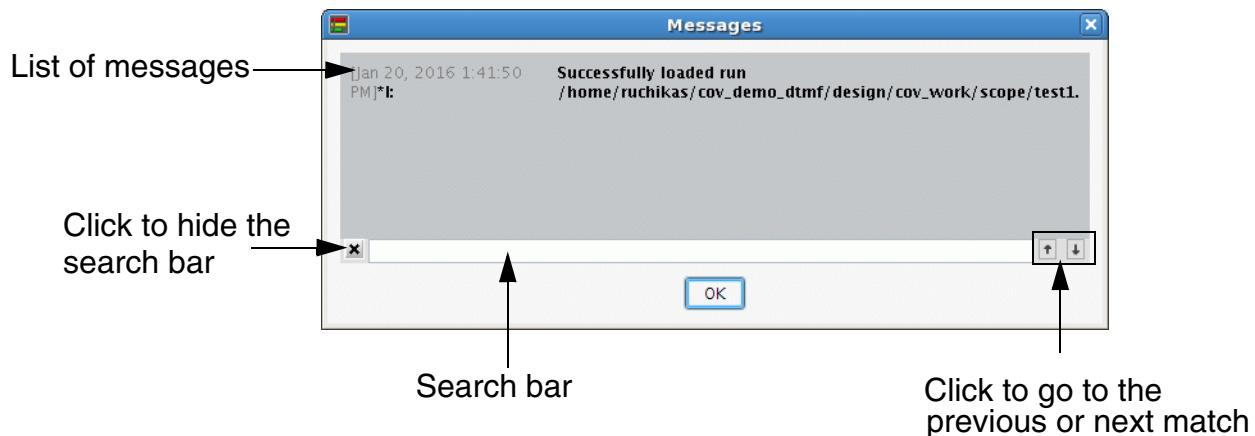


Later, as you load the runs, the metrics information is displayed in this area of the IMC window.

1.4.5 Messages Log

The IMC window displays *Messages* icon in the bottom left corner. When you click the *Messages* icon, the *Messages* dialog box is displayed, as shown in [Figure 1-28](#) on page 39.

Figure 1-28 Messages Log



The *Messages* dialog box shows the list of warnings or errors that occurred during the current IMC run.

In addition to viewing the message information, you can perform a search on the information displayed in the *Messages* dialog box.

To quickly search for a specific text, type the search text in the text box. As you type the text in the text box, the text gets highlighted in the messages area. You can use the *Up* and *Down* arrow keys to go to previous or next match. You can hide the search bar by clicking the *X* icon shown on the search bar. To reopen the search bar, press *Ctrl + F* keys.

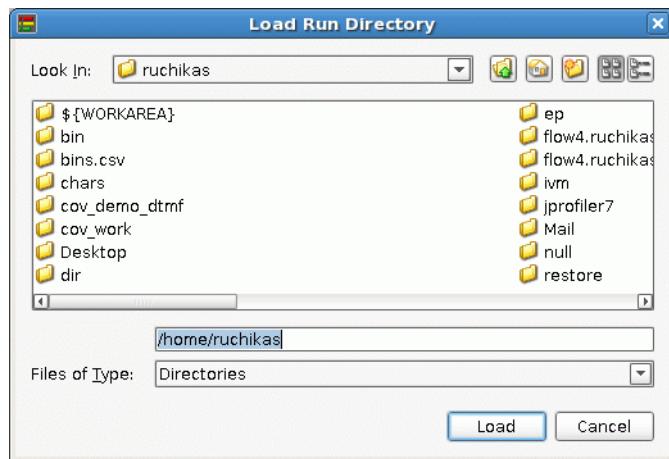
1.5 Loading Runs

To load a run:

1. Click the *File* menu and select *Load....*

The *Load Run Directory* dialog box is displayed, as shown in [Figure 1-29](#) on page 40.

Figure 1-29 Load Run Directory



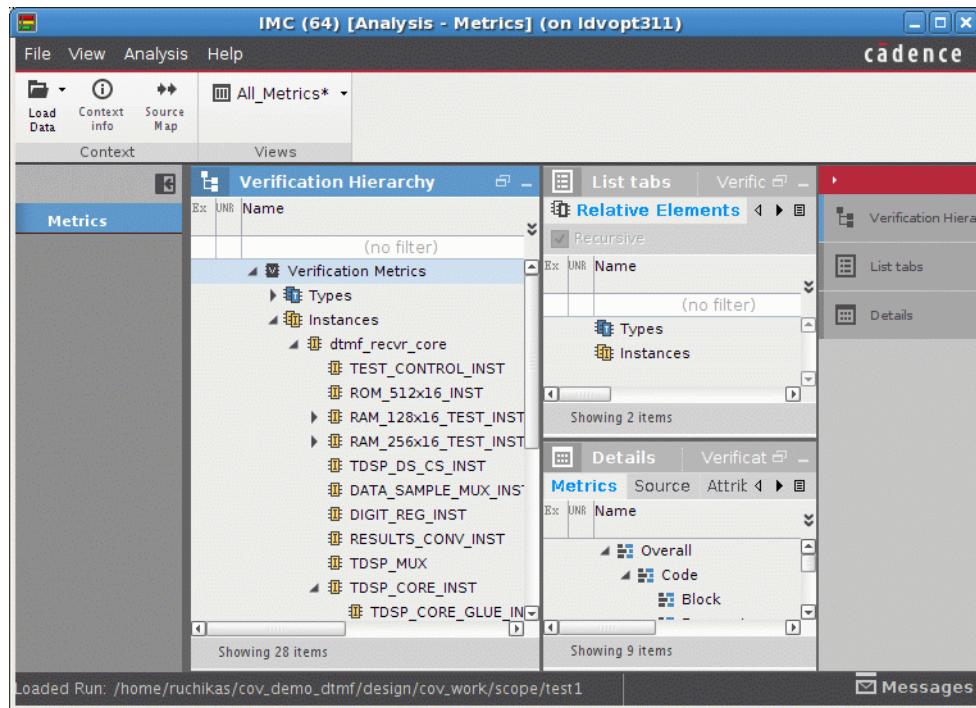
2. Navigate through the hierarchy and select the relevant run. For example, to load the run from the cov_work/scope/test1 directory, double-click cov_work, or select cov_work and click *Open*.

Note: In IMC, you can also load and analyze coverage coming from Palladium.

3. Double-click scope, or select scope and click *Open*.
4. Double-click test1, or select test1 and click *Load*.

The loaded run is shown in [Figure 1-30](#) on page 41.

Figure 1-30 IMC Window with Loaded Run



Alternatively, you can load a run by clicking the *Load* button in the toolbar.

Note: To load a run in batch mode, use the `load -run` command. For batch mode commands of IMC, see [IMC Command-Line Interface](#) on page 237.

Note: You cannot load more than one run in an IMC session. If a run is loaded and you want to load another run, then the earlier loaded run is automatically unloaded. If at the time of loading another run, the detailed analysis pages are open, then at the time of loading another run, a confirmation message, as shown in [Figure 1-31](#) on page 41 appears to indicate that loading of run will close all detailed analysis pages.

Figure 1-31 Reloading—Confirming Data Reload



You can click *OK* to confirm loading or click *Cancel* to dismiss loading. You can also select the *Don't ask me again (always approve)* check box to ensure that your choice (*OK* or *Cancel*) is saved and preserved for reuse in subsequent similar situations.

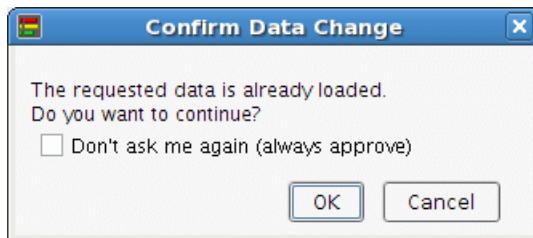
1.5.1 Reloading Run

To reload an already loaded run:

1. Click the *File* menu and select *Reload*.

A dialog box appears, as shown in [Figure 1-32](#) on page 42 to indicate that the data is already loaded and if you want to continue with reloading the data.

Figure 1-32 Reloading—Confirming Data Reload



2. You can:

- ❑ Click *OK* to confirm reloading.
- ❑ Click *Cancel* to dismiss reloading.

Note: You can also select the *Don't ask me again (always approve)* check box to ensure that your choice (*OK* or *Cancel*) is saved and preserved for reuse in subsequent similar situations.

Note: If at the time of reloading, the detailed analysis pages are open, then a confirmation message, as shown in [Figure 1-33](#) on page 43 appears to indicate that the run is already loaded and the detailed analysis pages will close when you reload the data.

Figure 1-33 Reloading—Confirming Data Reload



You can click *OK* to confirm reloading or click *Cancel* to dismiss reloading.

1.5.2 Unloading Run

To unload a loaded run:

1. Click the *File* menu and select *Unload*.

A dialog box appears, as shown in [Figure 1-34](#) on page 43 to confirm if you want to continue with unloading the data.

Figure 1-34 Unloading—Confirming Data Unload



2. You can:

- ❑ Click *OK* to confirm unloading.
- ❑ Click *Cancel* to dismiss unloading.

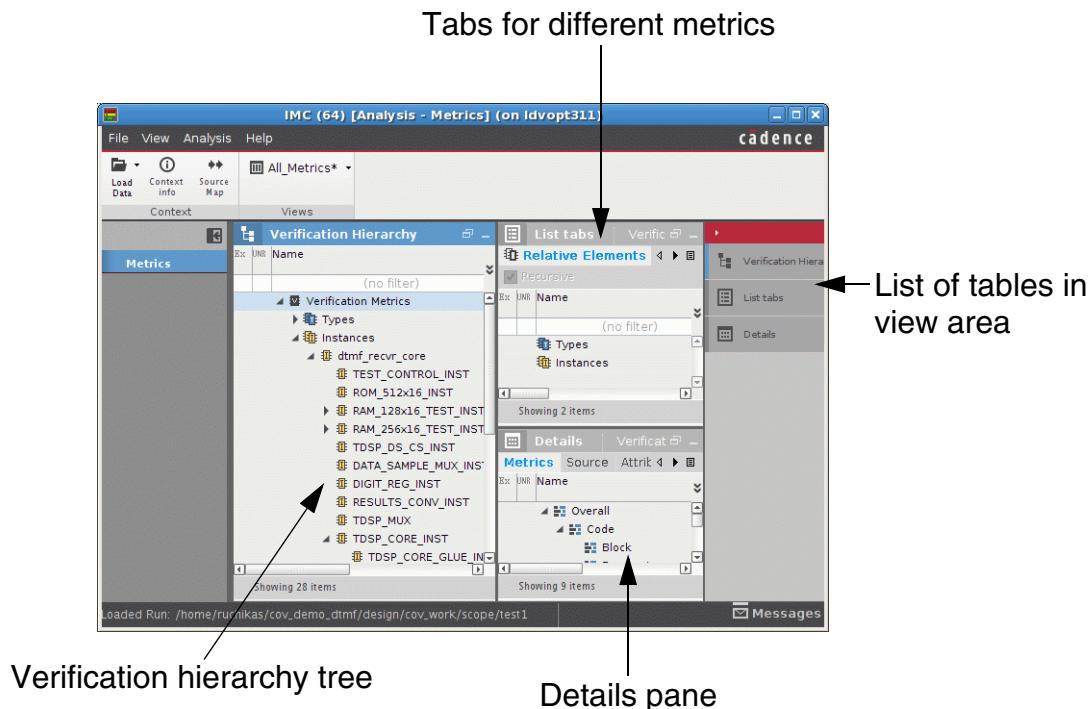
Note: You can also select the *Don't ask me again (always approve)* check box to ensure that your choice (*OK* or *Cancel*) is saved and preserved for reuse in subsequent similar situations.

1.6 Metrics Page

The *Metrics* page displays the overall grade, overall covered, uncovered, and local grades for different instances and types of the loaded run. Using the *Metrics* page, you can identify

the uncovered items that must be analyzed and targeted to improve coverage numbers. The *Metrics* page is shown in [Figure 1-35](#) on page 44.

Figure 1-35 Metrics Page



The *Metrics* page is divided into the following areas:

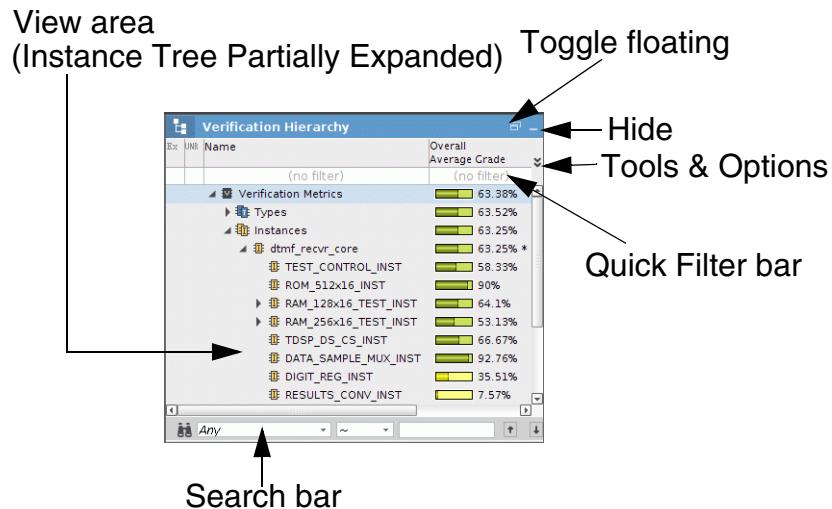
- [Verification Hierarchy Pane](#)
- [List tabs Pane](#)
- [Details Pane](#)

Note: The maximum number of items that can be selected in a table or a tree is limited to 1000.

1.6.1 Verification Hierarchy Pane

[Figure 1-36](#) on page 45 displays the *Verification Hierarchy* pane of the *Metrics* page. Using the *Verification Hierarchy* pane, you can navigate through the design hierarchy. The *Verification Hierarchy* pane by default, displays the instance hierarchy partially expanded, and information such as name of the instance or type, overall average grade, and overall covered grade.

Figure 1-36 Metrics Page—Verification Hierarchy Pane



The main components of the verification hierarchy pane are:

- [View Area](#)
- [Tools & Options](#)
- [Toggle Floating](#)
- [Hide](#)
- [Quick Filter Bar](#)
- [Search Bar](#)

1.6.1.1 View Area

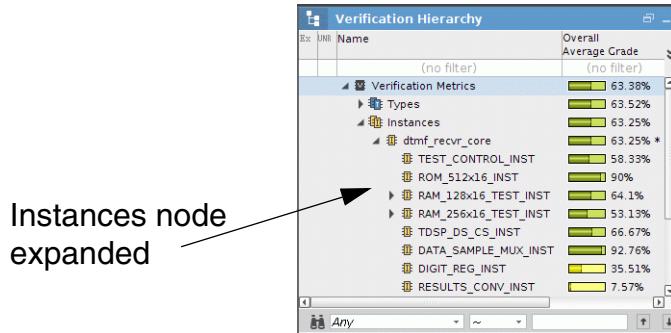
In the view area, at the top of the pane, an artificial node, Verification Metrics, is displayed.

Below the Verification Metrics node, there are following two nodes:

- *Types* node—Displays a list of all of the types scored within the design.
- *Instances* node—Displays a list of all of the instances scored within the design.

By default, when you load a run, the view area displays the *Instances* node expanded, as shown in [Figure 1-37](#) on page 46.

Figure 1-37 Instance Node Expanded



To expand the *Types* node, click the + symbol next to *Types*.

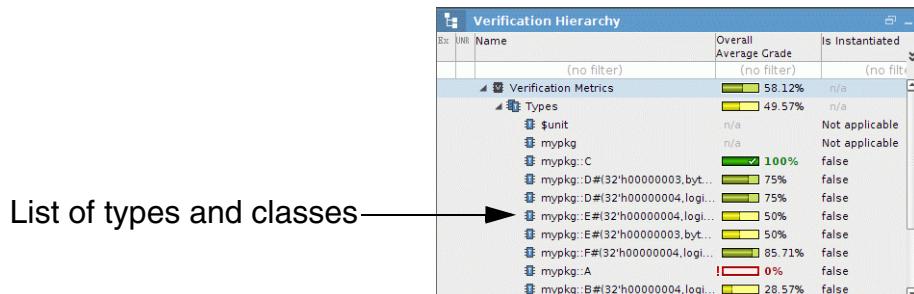
The *Types* node, when expanded, lists all the modules/types in the design.

Types Hierarchy in the case of SystemVerilog Classes

If type-based block coverage is enabled for a class, then the class is also listed in the *Types* hierarchy. For more details on enabling coverage, see the *ICC User Guide*.

Figure 1-38 on page 46 shows a *Types* node expanded with listed types and classes.

Figure 1-38 Class Listed



The type hierarchy has an entry for each of the class for which blocks are scored.

A type name for a class is displayed as:

```
(DeclarativeScopeName ::|_) { (DeclarativeScopeName ::|_) } className
```

where:

- `DeclarativeScopeName` is the name of the encapsulating container (package/compilation-unit/class/module/interface/program) in which the class or another encapsulating container is declared.
- `className` is the name of the class if the class is not parameterized. If the class is parameterized, `className` is the name of its unique specialization.

Note: The trailing separator after each encapsulating container (`DeclarativeScopeName`) can be `::` or `_` depending on its type. If the encapsulating container is a package or a class, then `::` is used as a separator. Else, an `_` is used as a separator.

Consider an example.

```
module mod1
  ...
  class c1
  ...
  class c2
  ...
endmodule
```

In this case, the type name for a class `c2` will be `mod1_c1::c2`.

Consider another example.

```
package P1
  ...
  class c1
  ...
  class c2
  ...
endpackage
```

In this case, the type name for a class `c2` will be `P1::c1::c2`.

Note: The similar naming is shown in ASCII as well as HTML reports. In addition, use the same naming mechanism in the `exclude` commands. For a few examples, see [Excluding All of the Blocks](#) on page 192.

The value of attribute *Is Instantiated* indicates if the class is instantiated or not. Possible values of this attribute are:

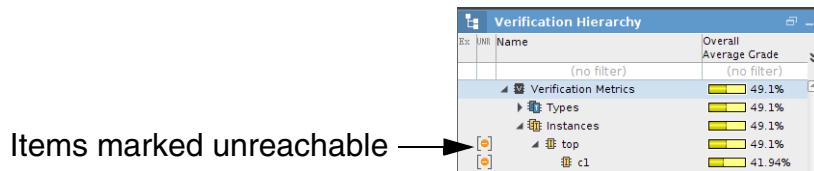
- `true` if the selected item is a class and it is instantiated.
- `false` if the selected item is a class and it is not instantiated.
- `Not applicable` if the selected item is a type and not a class.

Note: By default, the *Is Instantiated* attribute is not shown in the hierarchy pane. If required, you can add it to the hierarchy pane. For more details, see [Adding Columns on page 64](#). In addition, the attribute *Is Instantiated* will be shown in the reports only if it is part of the view.

UNR Indication in Hierarchy Tree

Sometimes, you might see an icon appearing in the *UNR* column in the verification hierarchy pane, as shown in [Figure 1-39 on page 48](#).

Figure 1-39 Items Marked Unreachable by IEV



An icon appears if the listed instance or type includes blocks, expressions, signals, or assertions that are marked unreachable (UNR) by *Incisive Enterprise Verifier* (IEV).

Note: The color of the icon and the tooltip indicates if the item was marked as fully unreachable or partially unreachable.

- A red color icon indicates fully unreachable. This means all the items within that instance or type are marked unreachable. A fully unreachable icon shows the tooltip as **Fully Unreachable**.
- An orange color icon indicates partially unreachable. This means that not all the items within that instance or type are marked unreachable. A partially unreachable icon shows the tooltip as **Partially Unreachable**.
- A totally gray icon indicates unreachable item that is marked excluded. This item was unreachable and later marked excluded in IMC. Such items shows the tooltip as **Excluded Unreachable**.

Note: Similar icons are also seen in the *Relative Elements* table of *List tabs* page and the *Metrics* tab page.

[Figure 1-40 on page 49](#) shows the attributes related to UNR items.

Figure 1-40 Attributes Related to UNR Items

Number of cumulative UNR entities

Details		c1	
Metrics	Source	Attributes	
Cal #	Name	Value	
	unr	(no filter)	
1	UNR	P_UNR	
	Overall UNR	7.0	
	Overall Local UNR	7.0	

Number of local UNR entities

UNR status
(F_UNR: Fully unreachable)
(P_UNR: Partially unreachable)
(E_UNR: Excluded unreachable)

Similar to other attributes, you can also add UNR related attributes in the *Verification Hierarchy* pane for analysis. For details on adding attributes, see [Adding Columns](#) on page 64.

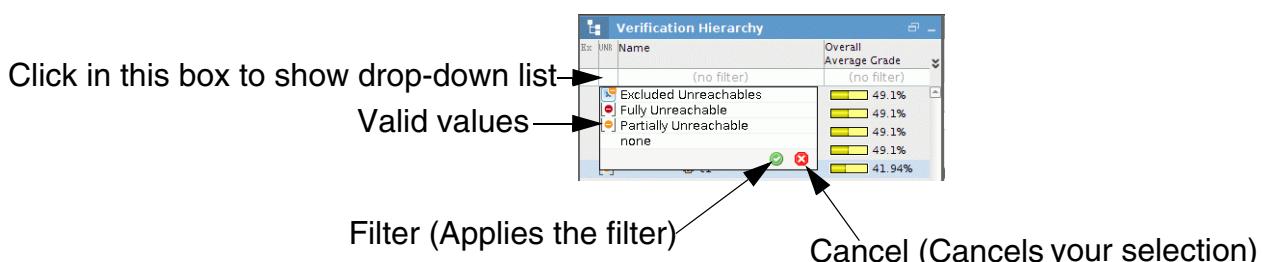
Filtering Items in UNR column (Verification Hierarchy Tree)

To apply filters to values in the *UNR* column:

1. Click in the Filter text box in the *UNR* column.

A drop-down is displayed with valid filter values, as shown in Figure 1-41 on page 49.

Figure 1-41 Filtering Values in UNR Column



2. Select an appropriate value from the list. You can select any of the following values:

- Excluded Unreachables—To show only the items that were unreachable and were later marked excluded in IMC.
- Fully Unreachable—To show only the items that were marked fully unreachable by IEV.
- Partially Unreachable —To show only the items that were marked partially unreachable by IEV.

- ❑ None—To show the items that are not marked as either fully unreachable or partially unreachable.

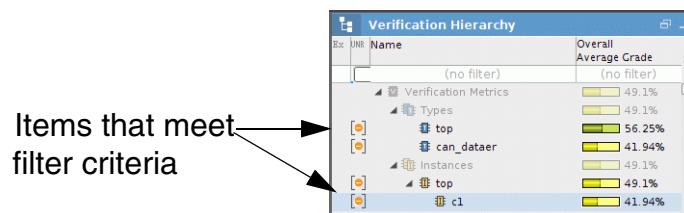
You can also select multiple values by holding the **Ctrl** key.

For example, to show only the items that were marked partially unreachable, select *Partially Unreachable* from the list.

3. Click the *Filter* icon to confirm the filtering action.

[Figure 1-42](#) on page 50 shows the filtered results.

Figure 1-42 Filtered Values in UNR Column



The filtered hierarchy tree shows following kinds of items:

- Items that meet the filter criteria. These items are shown as regular items.
- Items that partially meet the filter criteria. These items are shown as partially transparent. These are the items that do not meet the filter criteria completely, but appear because of their children.

Note: The items that do not meet the filter criteria are removed from the hierarchy tree.

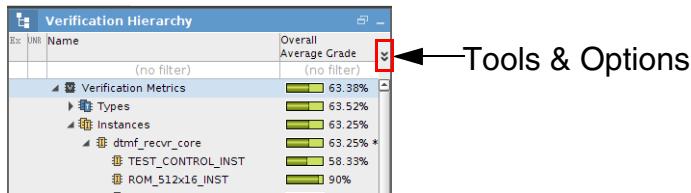
For details on the UNR flow and how items are marked UNR by IEV, see the *IEV User Guide*.

1.6.1.2 Tools & Options

The *Tools & Options* is a double arrow button which when clicked shows the options that allows you to add or remove columns in the view area, show or hide the search bar, and so on.

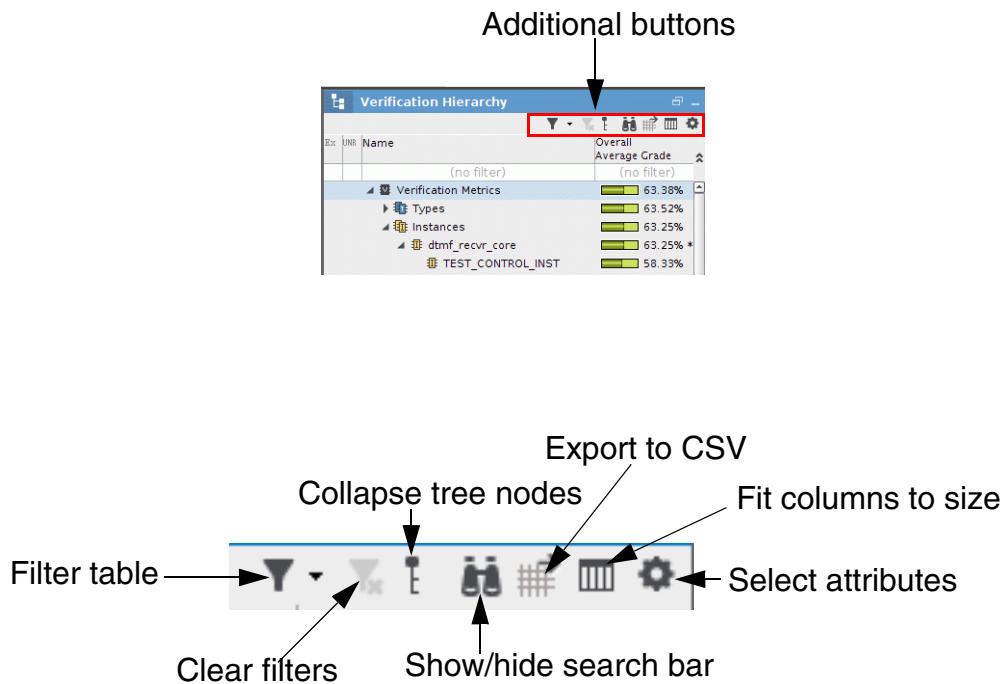
[Figure 1-43](#) on page 51 displays the *Tools & Options* button.

Figure 1-43 Tools & Options



When you click the Tools & Options button, additional buttons are displayed in the pane, as shown in [Figure 1-44](#) on page 51.

Figure 1-44 Tools & Options



The *Tools & Options* has following options:

- **Filter Table**—To apply quick filter, advanced filter, and show/hide quick filter bar. For more details, see [Filtering Table Data](#) on page 66.
- **Clear Filters**—To remove filters from all fields.
- **Collapse All**—To collapse the instance hierarchy tree and show only the Verification Metrics node.
- **Show Search Bar**—To show or hide the [Search Bar](#) shown at the bottom of the verification hierarchy pane.

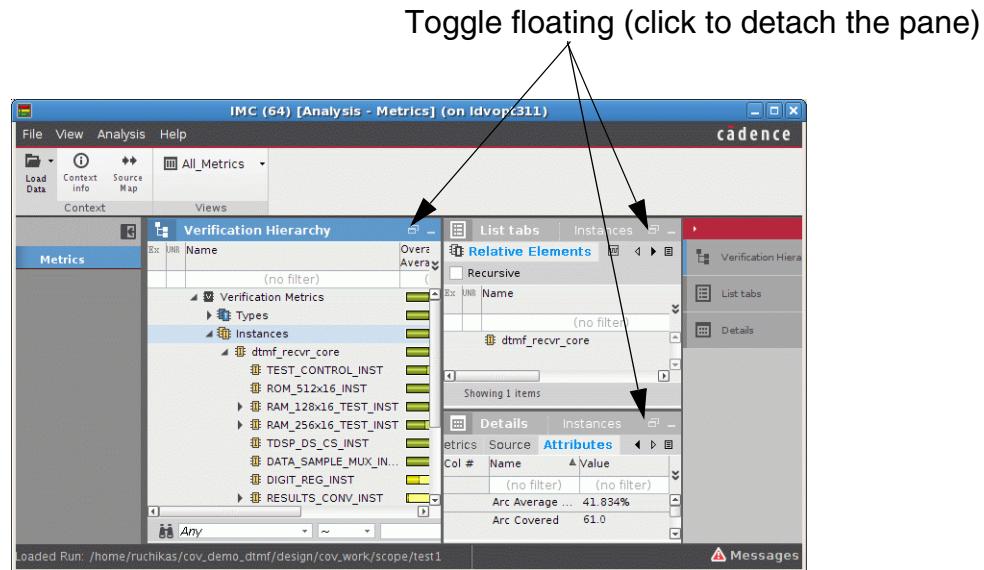
- Export to csv file —To export the metrics tree to the CSV file.
- Fit Columns to Size—To resize the width of the columns to its original default size.
- Select Attributes—To add or remove columns.

Note: Earlier, there was an option *Expand All*, which allowed expanding the instance hierarchy tree. However, IMC now allows loading unlimited number of entities to the tree, and therefore, it is unrealistic to allow expand all on trees as it can cause scalability issues.

1.6.1.3 Toggle Floating

[Figure 1-45](#) on page 52 displays the *Toggle floating* icon that appears on the title bar of different panes in IMC.

Figure 1-45 Toggle Floating

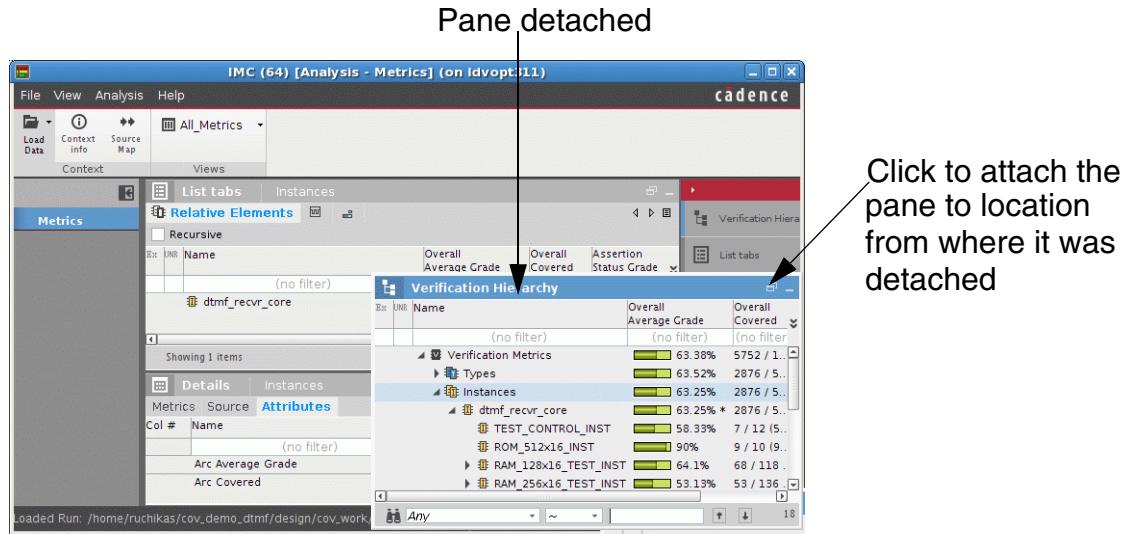


The *Toggle floating* icon is used to detach the pane from the view area and move it to a new location.

Note: You can also detach a pane by clicking on the title bar of the pane and then with the left-mouse button pressed, moving it to a new location.

After you click *Toggle floating* icon, the pane is detached, as shown in [Figure 1-46](#) on page 53.

Figure 1-46 Toggle Floating (Detached Pane)



The above figure shows the *Verification Hierarchy* pane detached. You can attach it back to the location from where it was detached by clicking the *Toggle floating* icon in the title bar of the pane.

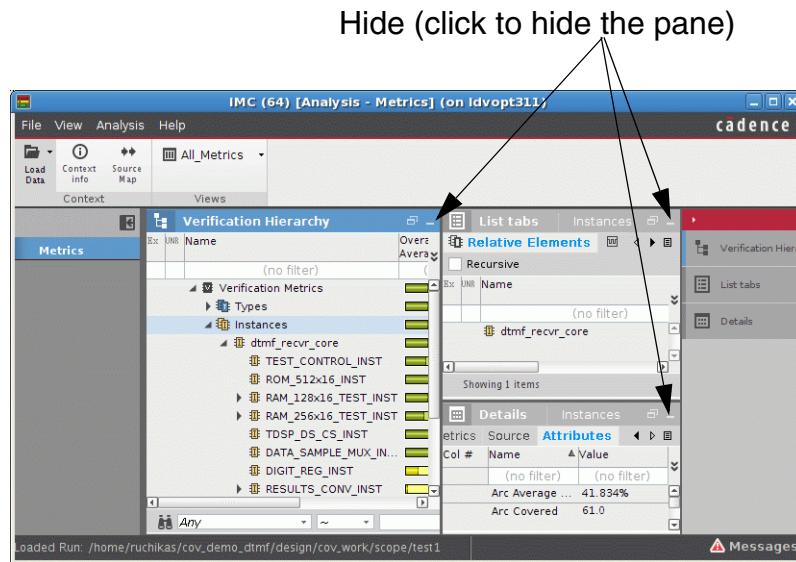
Note: You can also restore the detached pane back to its original location by selecting *Restore Current Layout* option in the *View* menu.

Note: The *Restore Current Layout* option sets the default layout for all the panes in all the opened views.

1.6.1.4 Hide

[Figure 1-47](#) on page 54 displays the *Hide* icon that appears on the title bar of different panes in IMC.

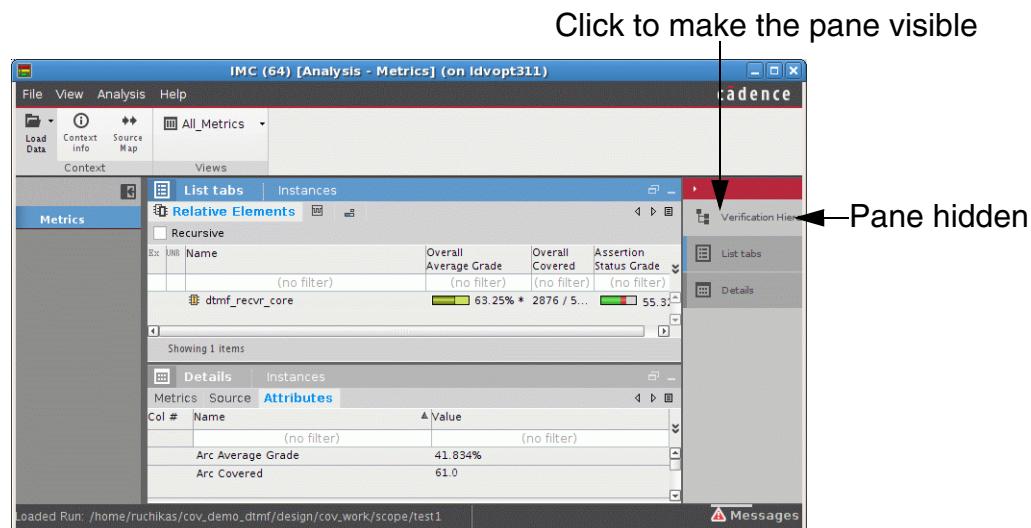
Figure 1-47 Hide



The *Hide* icon is used to hide the pane. For example, to hide the *Verification Hierarchy* pane, click the *Hide* icon in the title bar of the *Verification Hierarchy* pane.

[Figure 1-48](#) on page 54 shows the IMC screen with the hidden pane.

Figure 1-48 Toggle Auto-hide (Hidden Pane)



The above figure shows the *Verification Hierarchy* pane hidden. You can click the tab to restore the hidden pane to its original location.

Note: You can also restore the hidden pane to its original location by selecting *Restore Current Layout* option in the *View* menu.

Note: The *Restore Current Layout* option sets the default layout for all the panes in all the opened views.

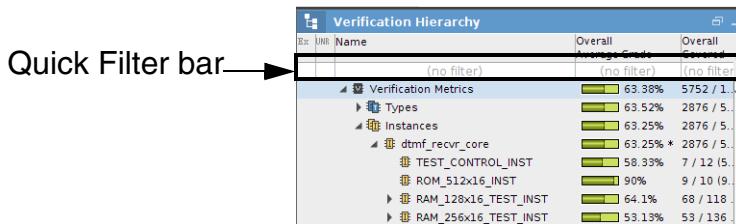


You can double-click on the title bar of a pane to maximize the pane.

1.6.1.5 Quick Filter Bar

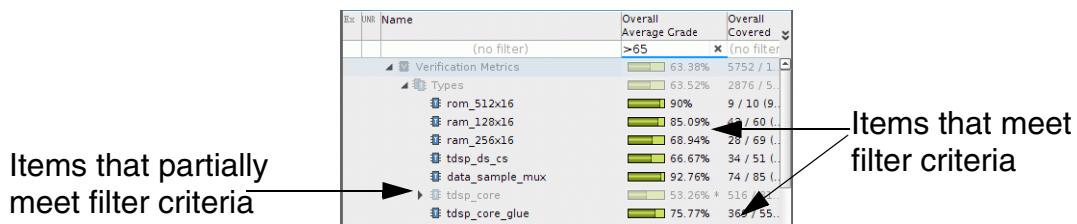
Figure 1-49 on page 55 displays the quick filter bar shown at the top of the verification hierarchy pane of the *Metrics* page. The quick filter bar helps you to filter the verification hierarchy pane based on the filter criteria.

Figure 1-49 Verification Hierarchy Pane—Quick Filter Bar



For example, to filter the verification hierarchy tree to show items for which *Overall Average Grade* is more than 65, specify *>65* in the filter row and press *Enter*. After you press *Enter*, the verification hierarchy tree is filtered, as shown in Figure 1-50 on page 55.

Figure 1-50 Verification Hierarchy Pane—Filtered



The filtered hierarchy tree shows following kinds of items:

- Items that meet the filter criteria. These items are shown as regular items.

- Items that partially meet the filter criteria. These items are shown as partially transparent. These are the items that do not meet the filter criteria completely, but appear because of their children.

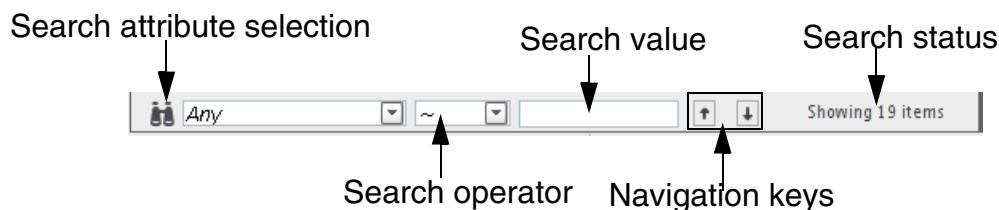
Note: The items that do not meet the filter criteria are removed from the hierarchy tree.

IMC also enables you to apply advanced filters. For more details, see [Advanced Filtering](#) on page 70.

1.6.1.6 Search Bar

[Figure 1-51](#) on page 56 displays the search bar shown at the bottom of the verification hierarchy pane of the *Metrics* page. The search bar helps you to quickly navigate through the instances or types that meet the specified search criteria.

Figure 1-51 Verification Hierarchy Pane—Search Bar



To navigate through the instance hierarchy based on a specified search criteria:

1. Select the attribute on which the search must be performed by clicking the search attribute selection drop-down. For example, to quickly navigate through the design hierarchy for instances where overall local grade is < 50 , select *Overall Local Grade* from the search attribute selection drop-down.

[Figure 1-52](#) on page 56 shows the search bar with *Overall Local Grade* selected.

Figure 1-52 Verification Hierarchy Pane—Search Bar (Attribute Selected)



2. Specify the search operator. For example, to set the criteria as *Overall Local Grade < 50*, select $<$ from the search operator drop-down.

[Figure 1-53](#) on page 57 shows the search bar with search operator specified as $<$.

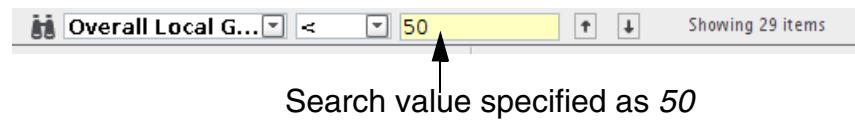
Figure 1-53 Verification Hierarchy Pane—Search Bar (Operator Selected)



3. Specify the search value and press Enter. For example, to set the criteria as Overall Local Grade <50, specify the value as 50 in the text box and press Enter.

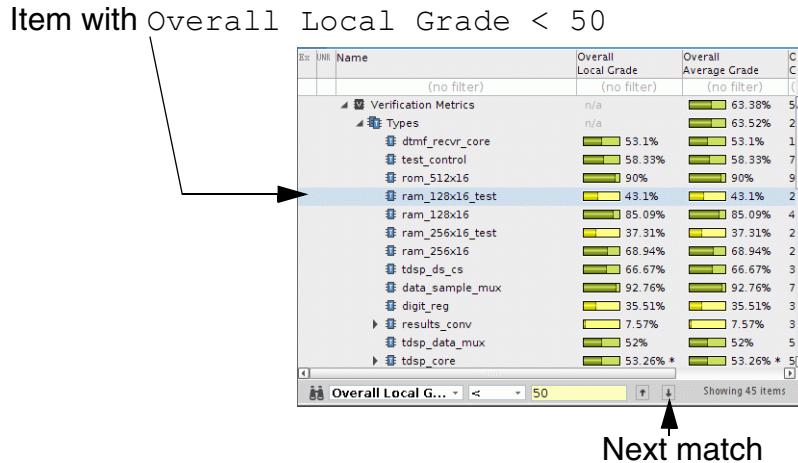
[Figure 1-54 on page 57](#) shows the search bar with search value specified as 50.

Figure 1-54 Verification Hierarchy Pane—Search Bar (Value Specified)



After specifying the search value, when you press the **Enter** key, the instance that meets the specified search criteria gets selected in the instance hierarchy, as shown in [Figure 1-55 on page 57](#).

Figure 1-55 Verification Hierarchy Pane



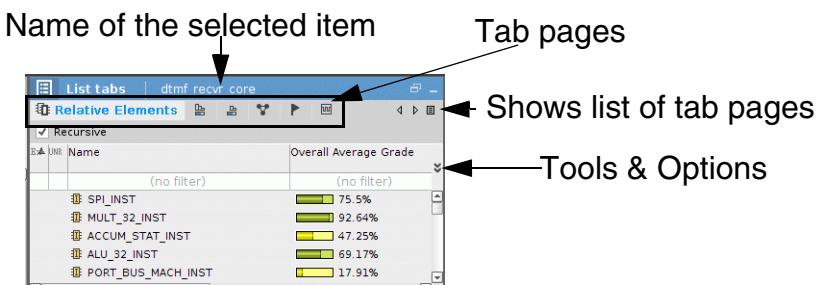
4. To navigate directly to the next instance that meets the search criteria, click the Next Match arrow key.

Similarly, you can quickly navigate to other instances, and also specify a different search criteria, as required.

1.6.2 List tabs Pane

[Figure 1-56 on page 58](#) displays the *List tabs* pane of the *Metrics* page. The *List tabs* pane has different tab pages to list relative elements, covergroups, FSMs, covergroup items, and assertions within the instance or type selected in the *Verification Hierarchy* pane of the *Metrics* page.

Figure 1-56 List tabs Pane



The *List tabs* pane has the following components:

- Tab pages to list relative elements, covergroups, FSMs, covergroup items, and assertions
- Tools & Options to add columns, remove columns, show/hide search bar, unfilter data, and unsort data
- Recursive check box to display or hide all objects within the selected instance

Note: When you select the recursive check box, an additional column is added to the table to display the enclosing entity for the listed items.

1.6.2.1 Tab pages

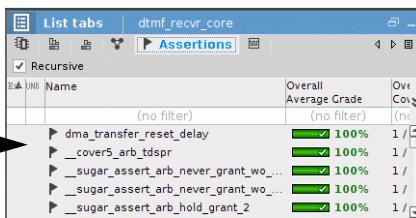
Using the different tab pages, you can list relative elements, covergroups, FSMs, covergroup items, and assertions within an instance or type selected in the *Verification Hierarchy* pane of the *Metrics* page. By default, the *Relative Elements* tab page is displayed, as shown in [Figure 1-56 on page 58](#).

To view a list of assertions within instance *TDSP_CORE_INST*:

1. Select *TDSP_CORE_INST* in the verification hierarchy pane.
2. Click the *Assertions* tab in the List tabs pane.

[Figure 1-57 on page 59](#) lists the assertions in the instance *TDSP_CORE_INST*.

Figure 1-57 List tabs—Assertions



List of assertions →

Name	Overall Average Grade	Overall Coverage
(no filter)	(no filter)	(no filter)
▶ dma_transfer_reset_delay	100%	1 /
▶ _cover5_arb_idspr	100%	1 /
▶ __sugar_assert_arb_never_grant_wo...	100%	1 /
▶ __sugar_assert_arb_never_grant_wo...	100%	1 /
▶ __sugar_assert_arb_hold_grant_2	100%	1 /

Similarly, you can list covergroups, covergroup items, FSMs, toggles, and child instances for the selected instance or type.

Note: By default, the *List tabs* pane of the *Metrics* page shows Relative Elements, Cover groups, Items, FSMs, Assertions, and Toggle tab pages. If the configuration option *Show extend metrics tree* is set, then only Relative Elements, Bins, and Toggle tab pages are shown in the *List tabs* pane. For more details on this option, see [Configuring Appearance Options](#) on page 80.

1.6.2.2 Tools & Options

The *Tools & Options* is a double arrow button which when clicked shows the options that allows you to add or remove columns in the view area, show or hide the search bar, and so on.

When you click the *Tools & Options* button, following additional buttons are displayed in the pane:

- Filter Table—To apply quick filter, advanced filter, and show/hide quick filter bar. For more details, see [Filtering Table Data](#) on page 66.
- Clear Filters—To remove filters from all fields.
- Return table to original order—To remove the sortings that were applied and show content in original order.
- Show Search Bar—To show or hide the [Search Bar](#) shown at the bottom of the pane.
- Export to csv file—To [export](#) the table data to the CSV file.
- Fit Columns to Size—To resize the width of the columns to its original default size.
- Select Attributes—To [add or remove](#) columns.

1.6.3 Details Pane

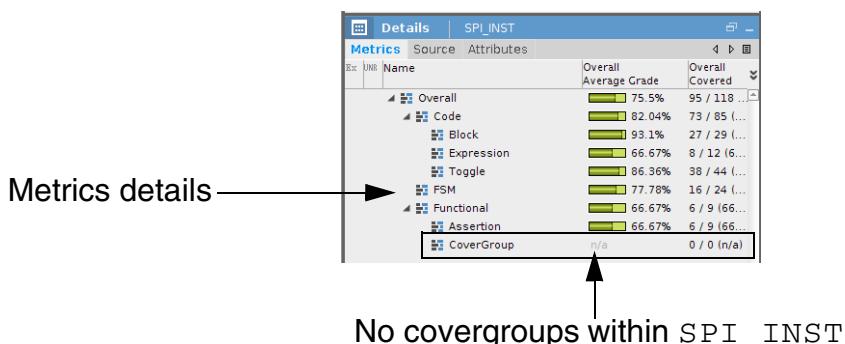
The *Details* pane shows additional information, such as the metrics scored, attributes, and source, for the item last selected in the *Verification Hierarchy* pane or the *List tabs* pane of the *Metrics* page. The *Details* pane has the following tabs:

- Metrics
- Attributes
- Source

1.6.3.1 Metrics

The *Metrics* tab displays the overall average grade, overall covered, and overall local covered details for each metric type relevant to the instance or type selected in the verification hierarchy pane. [Figure 1-58](#) on page 60 displays the *Metrics* tab for instance SPI_INST.

Figure 1-58 Details Pane—Metrics

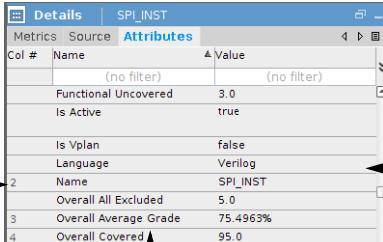


The value *n/a* for a metric type indicates that either the metric was not scored for that instance or type, or items corresponding to that metric type were not included in the selected instance or type. For example, *n/a* corresponding to CoverGroup in [Figure 1-58](#) on page 60 indicates that either there were no covergroups in instance SPI_INST, or covergroup coverage was not scored for SPI_INST.

1.6.3.2 Attributes

The *Attributes* tab displays the characteristics recorded for the item selected in the *Verification Hierarchy* pane or the *List tabs* pane of the *Metrics* page. [Figure 1-59](#) on page 61 displays the *Attributes* tab for an assertion.

Figure 1-59 Details Pane—Attributes



Col #	Name	Value
(no filter)	(no filter)	
Functional Uncovered	3.0	
Is Active	true	
Is Vplan	false	
Language	Verilog	
2 Name	SPI_INST	
Overall All Excluded	5.0	
3 Overall Average Grade	75.4963%	
4 Overall Covered	95.0	

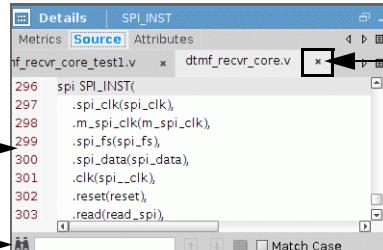
The attributes listed on the *Attributes* tab varies based on the selection made in the *Verification Hierarchy* pane or the *List tabs* pane of the *Metrics* page.

For more details on attributes associated with each item, see [List of Attributes](#) on page 419.

1.6.3.3 Source

The *Source* tab displays the actual source code for the item selected in verification hierarchy pane or the *List tabs* pane of the *Metrics* page. [Figure 1-60](#) on page 61 displays the *Source* tab for a assertion.

Figure 1-60 Details Pane—Source



Line Number	Code
296	spi SPI_INST;
297	.spi_clk(sp1_clk),
298	.m.spi_clk(m_sp1_clk),
299	.spi_fs(sp1_fs),
300	.spi_data(sp1_data),
301	.clk(sp1_clk),
302	.reset(reset),
303	.read(read_sp1)

The *Source* tab also displays the source of previously accessed items for easy access in different tabs. You can close the tab pages that show the source of previously accessed items by clicking the cross (X) symbol next to source file name. Using the search bar on the *Source* tab, you can quickly search for a specific text in the source code.

You can also right-click anywhere in the source code and select any of the following options:

- Copy —To copy the selected text.
- Copy Path—To copy full path of the source file to the system clipboard.

- Open in editor—To open the source file in the editor specified in the Configuration dialog box. For more details, see [Configuring General Options](#) on page 79.

Note: In IMC GUI, the *Source* tab can also show the source code from the compressed files provided the following is true:

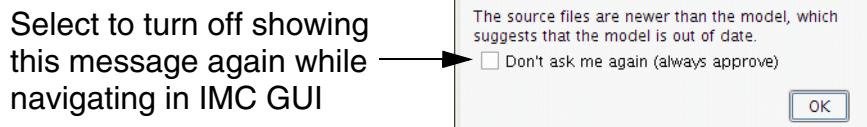
- The compressed files are of extensions .gz or .Z.
- Each compressed file contains only one file.
- Compressed files are used at the time of coverage data generation.



Currently, the batch mode of IMC cannot read the compressed files, and therefore, the *Source Code* column of the report of entities from the compressed files will be blank. This is also applicable for reports generated from GUI. The *Source Code* column of reports generated from GUI will be blank for entities from the compressed files.

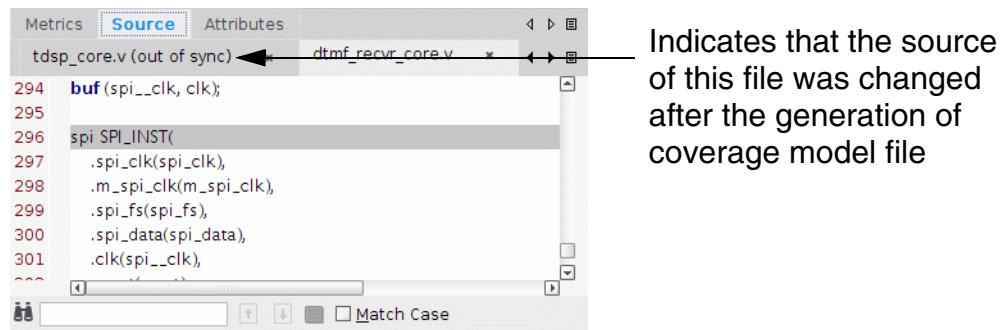
Note: If there is a change in the source code, after the generation of coverage model file, then while navigating in IMC GUI (when you happen to select the instance or type whose underlying source code has changed), a warning appears, as shown in [Figure 1-61](#) on page 62 to indicate that the source file is newer than the model file.

Figure 1-61 Warning to Indicate Change in Source Code



In addition, the *Source* tab shows an additional tab page indicating out of sync, as shown in [Figure 1-62](#) on page 63.

Figure 1-62 Additional Tab Page for Source Out of Sync



The text (out of sync) along with the source file name indicates that the source code of this file was changed after the generation of the coverage model file.

This helps the users to identify the source files that changed after the generation of coverage data.

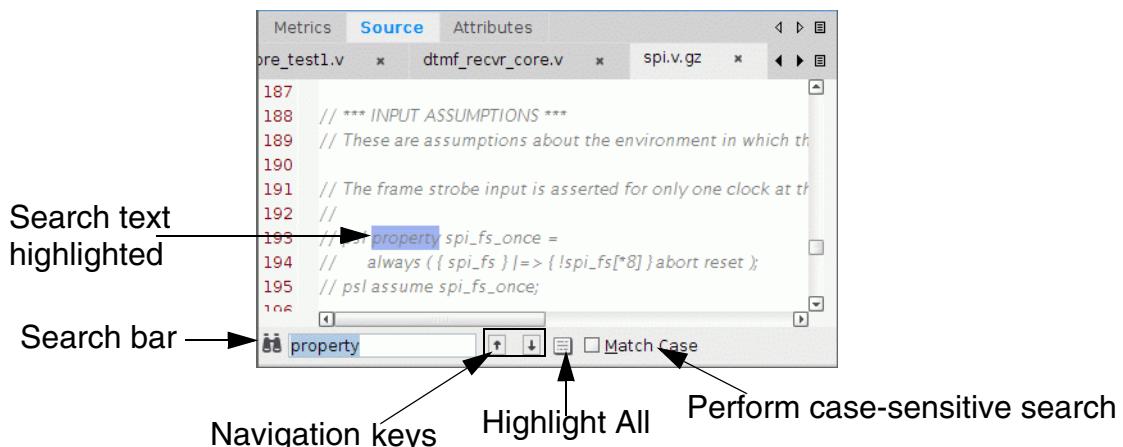
Performing a Search on the Source Code

[Figure 1-63 on page 63](#) displays the search bar available on the *Source* tab.

To quickly search for a specific text, type the search text in the text box. For example, type *property* in the text box.

As you type the text in the text box, the text gets highlighted in the source code.

Figure 1-63 Details Pane—Search Bar



You can:

- Use the *Up* and *Down* arrow keys to go to previous or next match.
- Click the *Highlight All* icon to highlight all occurrences of the search text in the source code.
- Select the *Match Case* check box to perform a search with the specified capitalization. For example, if you specify search text as `t_bit`, IMC will not search for `t_Bit` or `t_BIT`.

Note: The search applies to all open source files.

1.7 Managing Table Columns

You can resize, reorder, remove columns, and sort table data in different tables displayed in IMC GUI.

1.7.1 Resizing Columns

To resize a column, drag a separator in the header to a new location.

1.7.2 Reordering Columns

To reorder columns, drag the column header to its new location.

1.7.3 Sorting Column's Values

To sort on a column's values, click its column header. Click again to reverse the order.

Note: You can unsort table data by selecting *Unsort Data* from the Tools & Options drop-down.

1.7.4 Removing Columns

To remove a column, right-click the column header and select *Remove Attribute*.

Note: You can also remove columns using the *Attribute Selector* dialog box. For more details, see [Adding Columns](#) on page 64.

1.7.5 Adding Columns

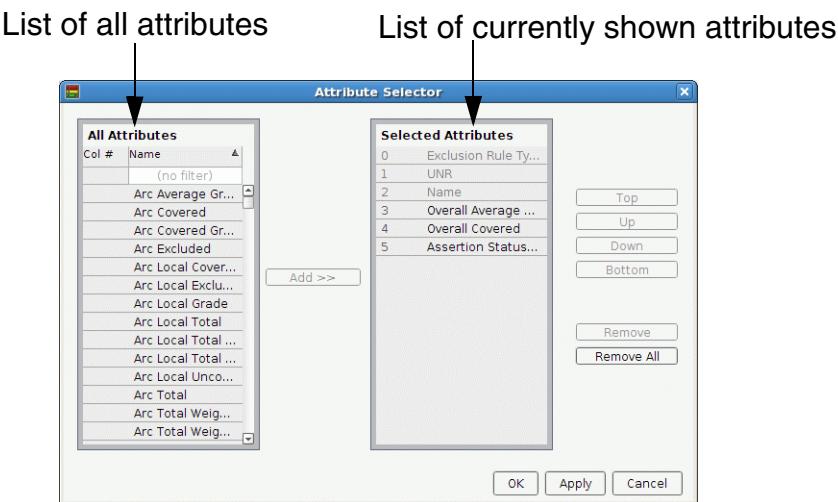
To add new columns:

1. Click the *Tools and Options* button and select *Select Attributes*.

Note: You can also right-click any of the column header in the table, and select *Select Attributes* from the pop-up menu.

2. The *Attribute Selector* dialog box opens, as shown in [Figure 1-64](#) on page 65. Select the attributes you want to display, and click *Add*.

Figure 1-64 Attribute Selector



In the *Attribute Selector* dialog box, you can add, remove, or define the placement of different attributes to be shown. The following buttons are available on this dialog box to perform these actions:

- Add—To add the selected attribute
- Top—To move the selected attribute to the top of the list
- Up—To move the selected attribute one step up in the list
- Down—To move the selected attribute one step down in the list
- Bottom—To move the selected attribute to the bottom of the list
- Remove—To remove the selected attribute
- Remove All—To remove all attributes present in the *Selected Attributes* list box.

Note: The attributes shown gray (*Exclusion Rule Type* and *Name*) cannot be removed and you cannot perform any of the above actions on these attributes.

Note: The movement of attributes due to actions, such as *Top* and *Up* applies only on attributes that are shown in black. As a result, when moving attributes using *Top* and *Up* buttons, the movement stops below the attributes *Exclusion Rule Type* and *Name*.

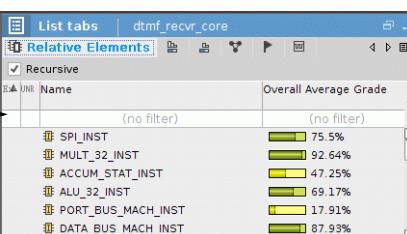
After you make required selections, the selected attributes are displayed in the table.

Note: Alternatively, you can open the *Attribute Selector* by right-clicking the column header and selecting *Select Attributes*.

1.8 Filtering Table Data

You can filter the data in tables to display only the required items. The row below the column header is the quick filter bar, which allows you to add the filtering criteria, as shown in [Figure 1-65 on page 66](#).

Figure 1-65 Filtering Data



Name	Overall Average Grade
(no filter)	(no filter)
SPI_INST	75.5%
MULT_32_INST	92.64%
ACCUM_STAT_INST	47.25%
ALU_32_INST	69.17%
PORT_BUS_MACH_INST	17.91%
DATA_BUS_MACH_INST	87.93%

This section covers the following topics:

- [Filtering Textual Values on page 66](#)
- [Filtering Numerical Values on page 67](#)
- [Filtering Values in Ex Column on page 68](#)
- [Advanced Filtering on page 70](#)

1.8.1 Filtering Textual Values

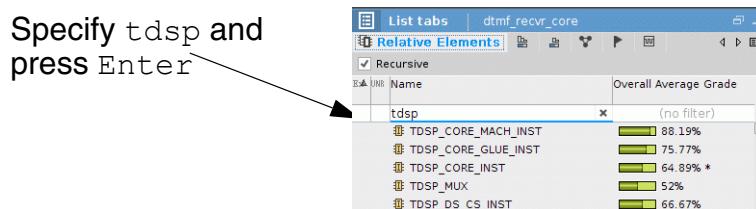
To filter textual values, specify the search string as any of the following:

- `~ value` to filter table data to show data that match the specified `value`. For example, `~trans` in *Name* column will show data where `trans` appears in the *Name*.

- `!~ value` to filter table data to show data that do not match the specified `value`. For example, `!~trans` in `Name` column will show data where `trans` does not appear in the `Name`.
- Note:** The default operator is `~`.
- `one_of value1,value2,valuen` to filter table data to show data that matches the comma separated values specified along with the `one_of` operator. For example, `one_of regs,dgb` in `Name` column will show data where `Name` is either `regs` or `dgb`.

For example, to filter the list of instances to display instances that include `tdsp` in instance names, specify `tdsp` in the filter text box, as shown in [Figure 1-66](#) on page 67, and press `Enter`.

Figure 1-66 Filter Textual Values



Notice that the list now displays only the instances that include `tdsp` in the instance name.

1.8.2 Filtering Numerical Values

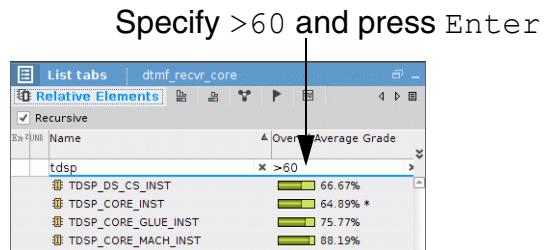
To filter numerical values, specify the search string as any of the following:

- `== value` to filter values that are equal to the specified `value`
- `!= value` to filter values that are not equal to the specified `value`
- `> value` to filter values that are greater than the specified `value`
- `>= value` to filter values that are greater than or equal to the specified `value`
- `< value` to filter values that are less than the specified `value`
- `<= value` to filter values that are less than or equal to the specified `value`

Note: The default operator is `==`.

For example, to filter `tdsp` instances where the overall average grade is more than 60, specify `>60` in the filter text box, as shown in [Figure 1-67](#) on page 68, and press `Enter`.

Figure 1-67 Filter Numerical Values



The only instances listed now are the ones with an overall average grade greater than 60. In addition, notice that if multiple filters are applied, then the result is the intersection of matches. In this case, only instances that include `tdsp` in the instance name with an overall grade of more than 60 are listed.

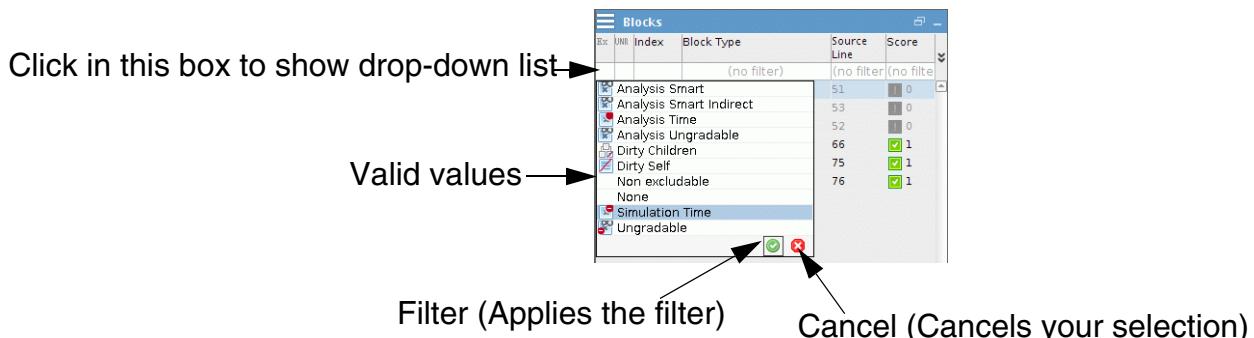
1.8.3 Filtering Values in Ex Column

To apply filters to values in the *Ex* column:

1. Click in the Filter text box in the *Ex* column.

A drop-down is displayed with valid filter values, as shown in Figure 1-68 on page 68.

Figure 1-68 Filtering Values in Ex Column



2. Select an appropriate value from the list. You can select any of the following values:

- Analysis Smart—To show only the items that were marked smart excluded at the time of analysis.
- Analysis Smart Indirect—To show only the items that were implicitly marked excluded because they were connected to the item that was smart excluded at the time of analysis.

- Analysis Time—To show only the items that were marked excluded at the time of analysis.
- Analysis Ungradable —To show only the items that were marked *Analysis Ungradable*.
- Dirty Children—To show only the parent entities that have children with orphan rules.
- Dirty Self—To show only the entities with orphan rules.
- Non excludable—To show only the items that cannot be marked excluded.
- None—To not apply any filter.
- Simulation Time—To show only the items that were marked excluded at the time of simulation run.
- Expression Ungradable —To show only the expressions that were marked ungradable by the simulator.
- Toggle Ungradable —To show only the toggles that were marked ungradable by the simulator.
- Ungradable —To show only the items that were marked ungradable by the simulator.

Note: A goal of 0 is automatically assigned to ungradable items and these items are ignored in coverage grading.

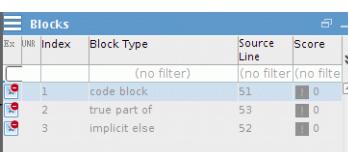
For more details on the above options, see [Refining Metrics Data](#) on page 167.

You can also select multiple values by holding the **Ctrl** key. For example, to show only the items which were marked exclude during simulation run, select *Simulation Time* from the list.

3. Click the *Filter* icon to confirm the filtering action.

After applying the filter, only the items that were marked exclude at the simulation time are shown in the table, as shown in [Figure 1-69](#) on page 69.

Figure 1-69 Filtered Values in Ex Column



Filtered values →

Ex	Index	Block Type	Source Line	Score
		(no filter)	(no filter)	(no filter)
1	1	code block	51	1 0
2	2	true part of	53	1 0
3	3	implicit else	52	1 0

Similarly, you can apply more filters based on your requirements.

1.8.4 Advanced Filtering

The advanced filtering feature enables you to add a complex filter to a table or to tree table.

It is useful when you want to apply more than one filter on the same attribute using AND or OR relationship and also useful in situations where you want to apply filters on attributes that are not displayed in the table at that time.

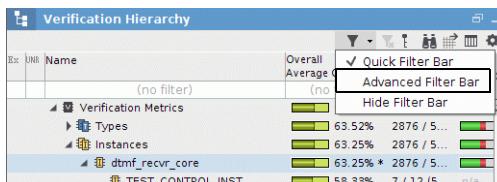
1.8.4.1 Displaying the Advanced Filter Bar

By default, the Quick filter bar is displayed in the different panes of IMC.

To apply advanced filters, you need to first open the Advanced filter bar. For this, perform the following steps:

1. Click the Tools & Options double arrow button. This will show additional buttons in the pane.
2. Now click the *Filter table* drop-down and select *Advanced Filter Bar* option, as shown in [Figure 1-70](#) on page 70.

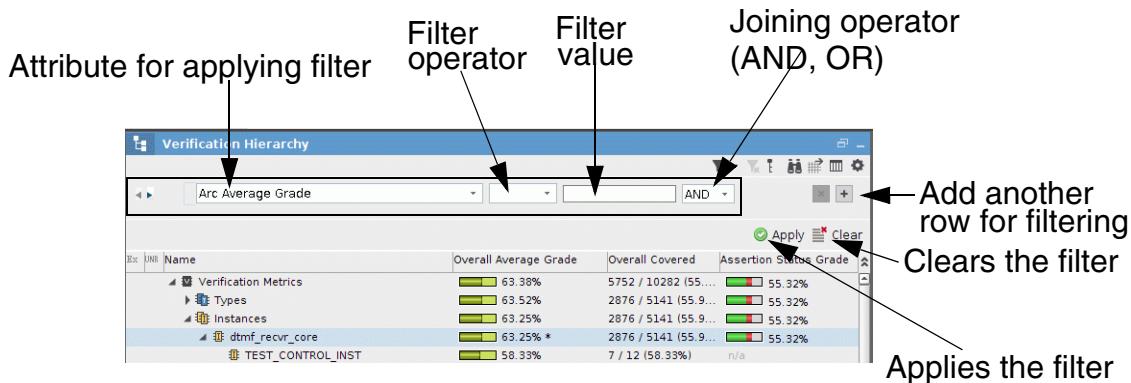
Figure 1-70 Advanced Filter



This will hide the Quick filter bar and show the Advanced filter bar.

[Figure 1-71](#) on page 71 shows the Advanced filter bar.

Figure 1-71 Advanced Filter Bar



1.8.4.2 Applying Advanced Filters

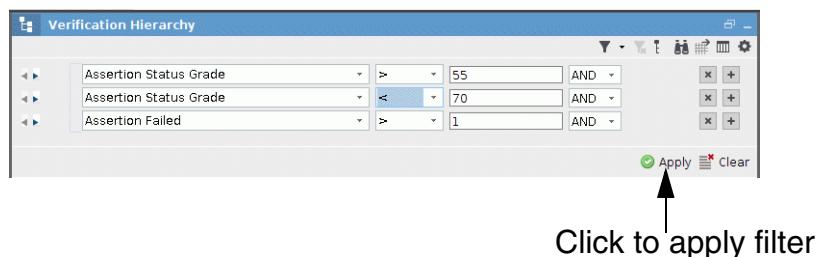
After displaying the Advanced filter bar, you can add complex filters, as required.

For example, you want to apply filter such that:

- *Assertion Status Grade* is >55 and <70 , or
- *Assertion Failed* >1 .

Figure 1-72 on page 71 shows the Advanced filter bar with required values.

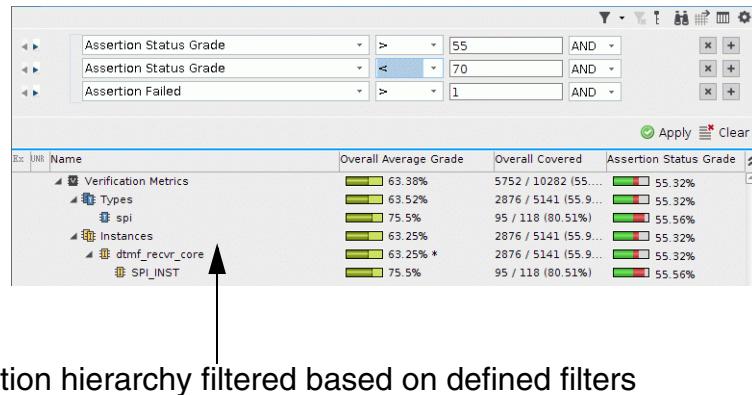
Figure 1-72 Advanced Filter Example



Click the *Filter* button to apply the filter.

Figure 1-73 on page 72 shows the Verification hierarchy filtered based on the defined filter.

Figure 1-73 Advanced Filter Example



Similarly, you can apply more filters based on your requirements.

1.9 Defining and Organizing Views

IMC allows you to define views such that you include only the attributes of your interest. You can also apply sorting and filtering on tables as per your requirement and save it as a view.

Note: After you save a view, it is available (listed in the *Views* drop-down) for all the analysis pages of same type. However, it is not applied automatically when you launch a new analysis page until you set the view as the default view. For example, you remove attributes, or add attributes on a block analysis page of an instance and save it as a view. When you open the block analysis page for another instance or type, then the saved view will be listed in the available views but it will not be set automatically. In case, you want a particular view to be used as the default view when you open a new analysis page, set that view as the default view.

This section covers the following topics:

- [Defining Views](#)
- [Saving Views](#)
- [Deleting Views](#)
- [Renaming Views](#)
- [Setting a Default View](#)
- [Organizing Views Under Different Folders](#)

1.9.1 Defining Views

For example, you can define a view named `MyView` such that:

- It shows only the *Overall Average Grade* in the *Verification Hierarchy* pane.

For this, you can remove the columns other than *Overall Average Grade* by right-clicking the column header and selecting *Remove Attribute*. For more details on removing columns, see [Removing Columns](#) on page 64.

- It shows only the instances that include `test` in the instance name.

For this, you can filter the data in the *Relative Elements* table to show only the instances that include `test` in the instance name. For more details on filtering, see [Filtering Table Data](#) on page 66.

After defining a view you must save it so that is available to you in later IMC runs.

1.9.2 Saving Views

To save a view:

1. Click the *All_Metrics* drop-down in the *Views* toolbar.
2. Select *Save As View* from the drop-down menu.

The *Save As View* dialog box is displayed, as shown in [Figure 1-74](#) on page 73.

Figure 1-74 Save View



3. Specify the name of the view and click *OK*. For example, to save the view as `MyView`, specify `MyView` in the text box and click *OK*.

The view is created and listed, as shown in [Figure 1-75](#) on page 74.

Figure 1-75 Saved View



The views are stored as an XML file in a directory named `.imc/PageViews`. The default location is `~/.imc/PageViews`. The views you save are available in the next IMC run. As a result, it saves your time as you do not need to create views in each IMC invocation.

Note: It is possible to share IMC views between users. Views can be generated by one user and his/her `.imc` directory can then be copied to the PROJECT area. Environment variable `IMC_PAGE_VIEWS_DIR` can be set to the directory containing `UserDefinedPageViews` file and all users that have the variable will be able to use the views that were created.

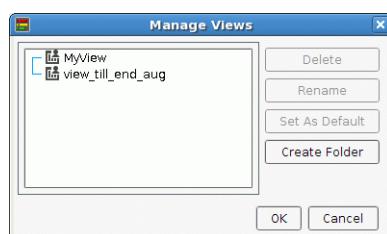
1.9.3 Deleting Views

To delete a view:

1. Click the drop-down in the *Views* toolbar.
2. Select *Manage Views* from the drop-down menu.

The *Manage Views* dialog box is displayed, as shown in [Figure 1-76](#) on page 74.

Figure 1-76 Manage Views



3. Select the view you want to delete and click *Delete*.
4. Click *OK*.

The selected view is then deleted from the list of views.

1.9.4 Renaming Views

To rename a view:

1. Click the drop-down in the *Views* toolbar.
2. Select *Manage Views* from the drop-down menu.
3. Select the view you want to rename and click *Rename*. For example, select *MyView* and click *Rename*.

The *Rename* dialog box is displayed.

4. Specify the new name for the view and click *OK*.

The new name shows in the *Manage Views* dialog box.

5. Click *OK*.

1.9.5 Setting a Default View

Using IMC, you can set a particular view as the default view. The view that you set as the default view in one of the IMC runs is automatically set in subsequent IMC runs.

For example, you set *My_tests_view* as the default view and exit out of IMC. Later, when you again launch IMC, by default, *My_tests_view* will already be set.

You can set a view as the default view using any of the following methods:

- From the All_Metrics drop-down in the Views toolbar
- From the Manage Views Dialog Box

1.9.5.1 From the All_Metrics drop-down in the Views toolbar

To set a view as the default view:

1. Launch the view that you want to set as the default view. For this, select that view from the drop-down in the *Views* toolbar.

2. After the view is launched, select *Set as Default* from the *All_Metrics* drop-down in the *Views* toolbar.

This will set the selected view as the default view.

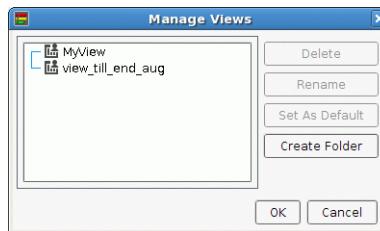
1.9.5.2 From the Manage Views Dialog Box

To set a view as the default view:

1. Click the drop-down in the *Views* toolbar.
2. Select *Manage Views* from the drop-down menu.

The *Manage Views* dialog box is displayed, as shown in [Figure 1-77](#) on page 76.

Figure 1-77 Manage Views (Set Default View)



3. Select the view that you want to set as the default view.
4. Click the *Set As Default* button.
5. Click *OK*.

This will set the selected view as the default view.

1.9.6 Organizing Views Under Different Folders

Using IMC, you can organize views under different folders. By default, all the views are listed at one level. In the case of multiple views, it is difficult to find the required view. You can create separate folders and organize your views in different folders.

To organize views in different folders:

1. Click the drop-down in the *Views* toolbar.
2. Select *Manage Views* from the drop-down menu.

The *Manage Views* dialog box is displayed.

3. Click *Create Folder*.

The *Create Folder* dialog box is displayed.

4. Specify the name of the folder in the text box and click *OK*.

The folder is created and shown in the *Manage Views* dialog box.

5. You can now move views to the folder. Select the view you want to move. Drag the view and drop it in the required folder.

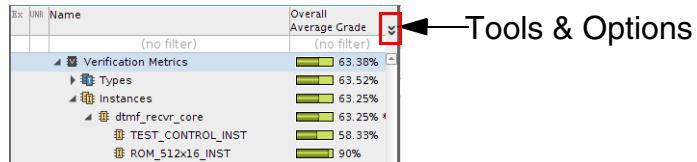
6. Click *OK* to close the *Manage Views* dialog box.

1.10 Exporting Metrics Tree to a CSV File

To export metrics tree to a CSV file, perform the following steps:

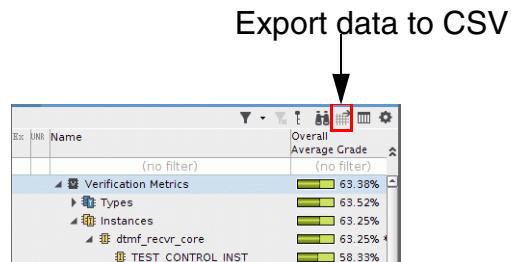
1. Click the *Tools and Options* double arrow, as shown in [Figure 1-78](#) on page 77.

Figure 1-78 Export to CSV File



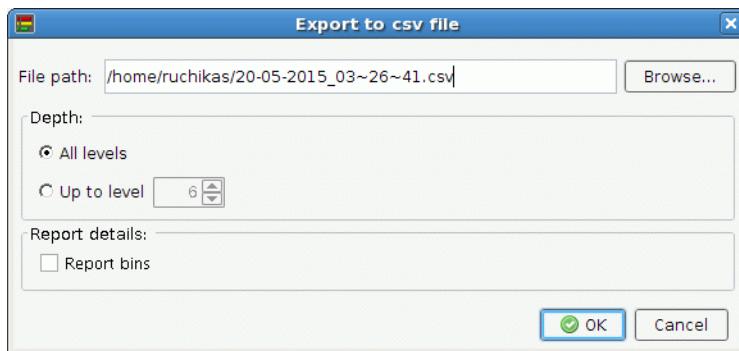
2. Select *Export data to csv file* from list of advanced buttons, as shown in [Figure 1-79](#) on page 77.

Figure 1-79 Export to CSV File



3. The *Export to csv file* dialog box appears, as shown in [Figure 1-80](#) on page 78.

Figure 1-80 Export to CSV File



4. In the *File path* field, you can specify the location and name of the file where the exported file must be saved. By default, it shows the location from which IMC was invoked and the file name is shown as <timestamp>.csv. You can change it, as required.
5. Next, specify the levels of hierarchy that must be exported to CSV file. You can select any of the following:
 - All levels — To export all the levels of hierarchy. By default, this option is selected.
 - Up to level — To specify the number of levels to be exported. By default, 6 is specified in this field. You can increase or decrease the levels, as required.
6. To include cover bins in the CSV file, select the *Report bins* check box.
7. Click *OK*.

After you click *OK*, the metrics tree is exported to the CSV file at the specified location.

Note: The data included in the exported CSV file is based on the view selected at the time of generating the report and also the filters, grouping, and sorting applied at the time of exporting the CSV.

Note: If an attribute's value is n/a in GUI, then it is taken as an empty string in the exported CSV file.

You can also export the metrics tree from the command-line interface of IMC. For more details, see [Exporting Metrics Data to CSV File](#).

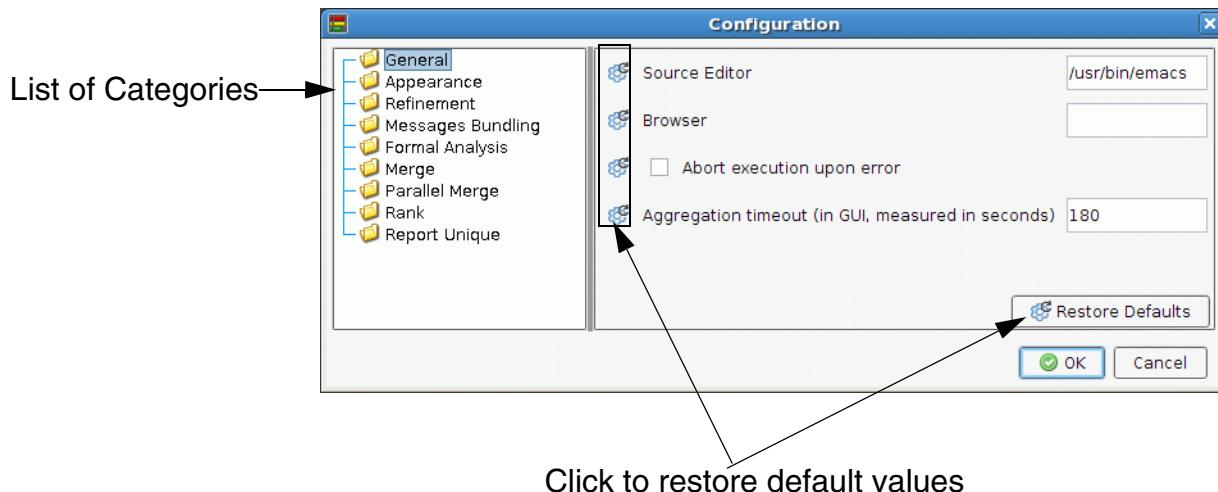
1.11 Configuring Configurable Items

The *Configuration* dialog box allows you to configure general options and options related to appearance, refinement, and message logs.

You can invoke the *Configuration* dialog box by selecting *Configuration* from the *View* menu.

[Figure 1-81](#) on page 79 displays the *Configuration* dialog box.

Figure 1-81 Configuration



You can click any of the categories in the left pane and its related options are shown in the right pane of the *Configuration* dialog box.

Note: For the numeric fields in the *Configuration* dialog box, there is a valid range of values that can be specified. In case you specify a value outside the range of valid values, the field becomes red to indicate that an invalid value is being added and tool by default, assigns the default value for that particular field to it.

Note: You can also configure items in the command line interface of IMC using the `config` command. For more details, see [Configuring Configurable Items](#) on page 252.

1.11.1 Configuring General Options

When you invoke the *Configuration* dialog box, the *General* options are shown by default.

The *General* category shows the following option:

- **Source Editor**—To specify an alternate source editor for opening source files. By default, `emacs` is used as the source editor for opening source files.

Note: Unlike `emacs` and `gvim`, the `vim` editor does not open a new terminal. As a result, if you want to use `vim` as the source editor, specify `xterm -e /usr/bin/vim` in the *Source Editor* field.

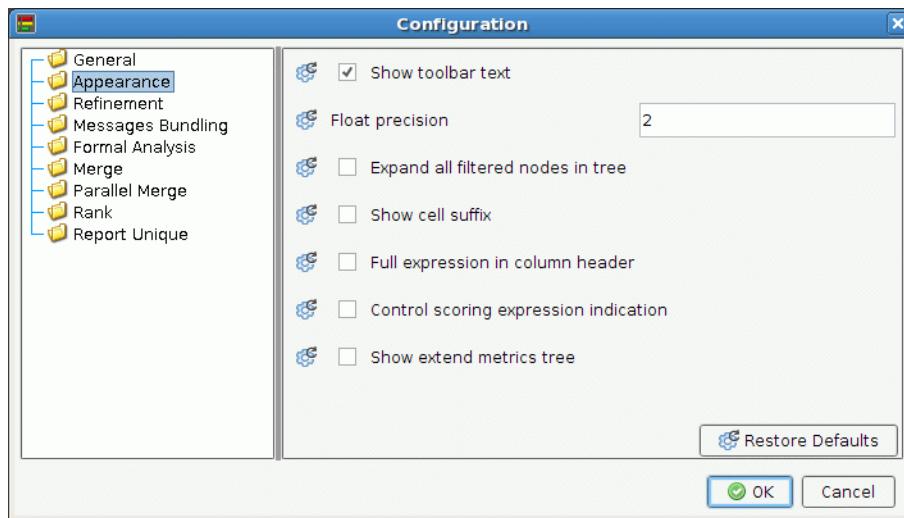
- Browser—To specify the path of the HTML browser. For example, you can specify the path as: /usr/bin/firefox.
- Abort execution upon error — By default, this check box is not selected. This indicates that execution is not aborted even if a command in the TCL file specified with the `-exec` option fails. When this check box is selected, then if any of the commands in this TCL file specified with the `-exec` option fails, the tool terminates without executing the remaining commands.
- Aggregation timeout (In GUI, measured in seconds)— To specify the aggregation timeout limit. By default, 180 seconds is specified. This indicates the aggregation calculation stops after 180 seconds. You can modify this limit, as required.

1.11.2 Configuring Appearance Options

To view and configure appearance related options, click *Appearance* in the left pane in the *Configuration* dialog box.

[Figure 1-82](#) on page 80 displays the *Configuration* dialog box with appearance related options.

Figure 1-82 Configuration -- Appearance Options



The *Appearance* category shows the following options:

- Show toolbar text—To enable or disable showing of text in the toolbar. By default, in each toolbar, just below the icon button, text is also written. To hide the text that appear below the icon buttons on toolbar, clear the *Show toolbar text* check box.

- **Float precision**—To specify the maximum number of decimal places that you want to see in the attributes that show values as decimals. By default, you see values with maximum 2 decimal places. You can increase or decrease the precision, as required. Valid range of values is 0 to 10. In case you specify a value outside the range of valid values, tool by default, assign the default value (which in this case is 2) to it.
- **Expand all filtered nodes in tree**—To enable or disable expanding of filtered nodes in the verification hierarchy tree. By default, expanding is turned off. To enable expanding of filtered nodes in the verification hierarchy tree select the *Expand all filtered nodes in tree* check box.
- **Show cell suffix**—To enable or disable showing the suffix of the attributes of type string. This option actually right justifies the content in the column. By default, this option is turned off.
- **Full expression in header column**—To enable or disable showing the expression in the column header in the Expression analysis page. By default, it shows the term numbers, such as, T1, T2, and so on. If you enable this option, then instead of T1, T2, the corresponding expression from the source code is shown.
- **Control scoring expression indication**—To enable or disable showing a * notation in front of the controlling terms in all expression coverage tables. By default, a * notation is not shown for identification of controlling terms. After you enable this option, a * notation appears next to the controlling terms in expression tables in GUI, HTML reports, and ASCII reports.

Note: This feature is enabled only if `set_expr_scoring -control` CCF command is used during simulation run. For more details on this command, see the *ICC User Guide*.

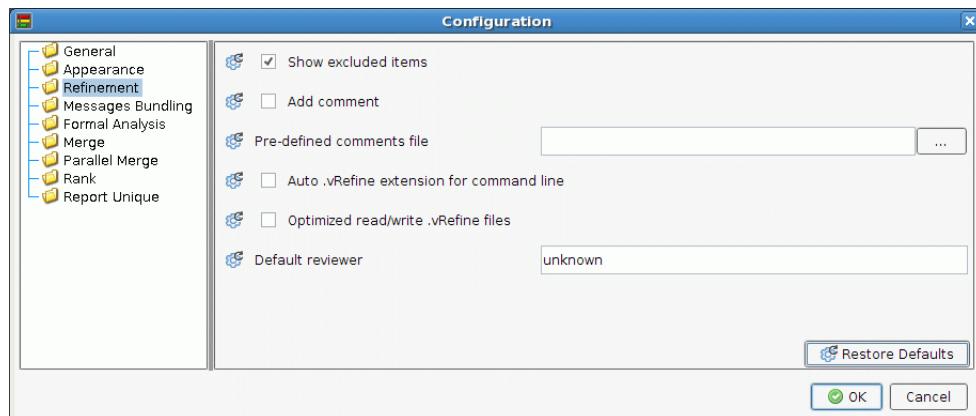
- **Show extend metrics tree**—To enable or disable showing extended metrics tree. By default, only instances and types are listed in the metrics hierarchy tree. After you enable this option, covergroups, cover items, FSMs, and assertions are also listed in the metrics hierarchy tree. In addition, only Relative Elements, Bins, and Toggle tab pages are shown in the *List tabs* pane of the *Metrics* page. By default, the *List tabs* pane of the *Metrics* page shows Relative Elements, Cover groups, Items, FSMs, Assertions, and Toggle tab pages.

1.11.3 Configuring Refinement Options

To view and configure refinement related options, click *Refinement* in the left pane in the *Configuration* dialog box.

Figure 1-83 on page 82 displays the *Configuration* dialog box with refinement related options.

Figure 1-83 Configuration -- Refinement Options



The *Refinement* category shows the following options:

- Show excluded items—To enable or disable showing of excluded items in the hierarchy. By default, excluded items appear in the hierarchy (though the coverage for that item is excluded from overall coverage). You can hide the excluded items by clearing the *Show excluded items* check box. For more details, see [Hide Excluded Items](#) on page 184.
- Add comment —To enable or disable adding comments at the time of exclusion. By default, the *Add comment* check box is not selected. This indicates that you will not be prompted to add comments at the time of exclusion. After you select the *Add comment* check box, you will be prompted to add comments at the time of exclusion. A dialog box will be displayed which will prompt you to add comments. For more details, see [Adding Comments at the Time of Exclusion](#) on page 180.
- Pre-defined comments file—To enable adding comments to exclusion from a pre-defined list of comments. A comments file is created as a .csv file in the following format:

```
NAME,COMMENT
<comment_name>,<comment_text>
<comment_name>,<comment_text>
```

You can specify the path to the comments file in the *Pre-defined comments file* field. After this, at the time of applying exclusion, (in the *Exclusion* dialog box) you will see an option that will allow you to select the comment from the pre-defined comments that were defined in the comments file. For more details, see [Adding Comments at the Time of Exclusion](#) on page 180.

- Auto .vRefine extension for command line —To automatically add .vRefine as the file extension if the user has not explicitly specified it at the time of saving the refinement file.

Note: This option affects only the batch mode. In GUI, the .vRefine extension is always added automatically, even if the user has not explicitly specified it.

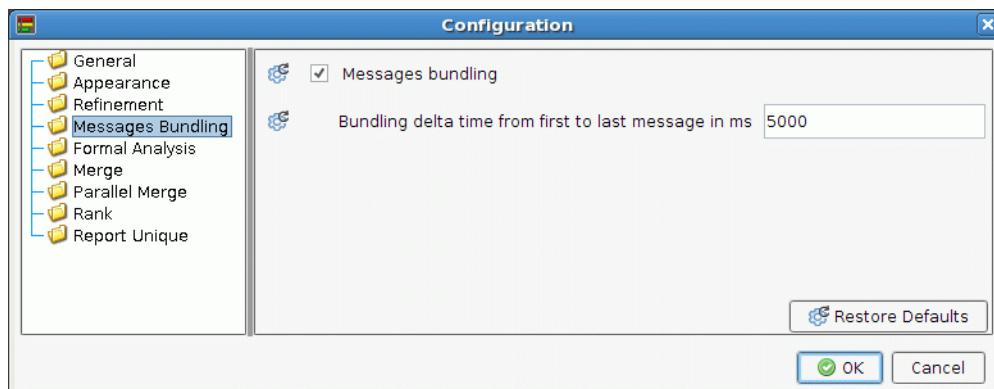
- Optimized read/write .vRefine files —To optimize the tool performance for read/write operation on a large .vRefine file. By default, this option is not selected. It is recommended that if the refinement file is relatively large, then you select this checkbox to optimize tool performance.
- Default reviewer—To specify the default reviewer for adding comments. By default, the reviewer is specified as unknown. You can change the reviewer, as required. The name you specify here will appear in the Exclusion dialog box when you want to add comments at the time of exclusion. For more details, see [Adding Comments at the Time of Exclusion](#) on page 180.

1.11.4 Configuring Messages Bundling Options

To view and configure message bundling related options, click *Messages Bundling* in the left pane in the *Configuration* dialog box.

[Figure 1-84](#) on page 83 displays the *Configuration* dialog box with message bundling related options.

Figure 1-84 Configuration -- Messages Bundling Options



The *Messages Bundling* category shows the following options:

- Messages bundling—To bundle messages of same type (in a given period of time) into a single message. By default, message bundling is turned on.
- Bundling delta time from first to last message in ms —To specify the maximum gap between messages for them to be bundled.

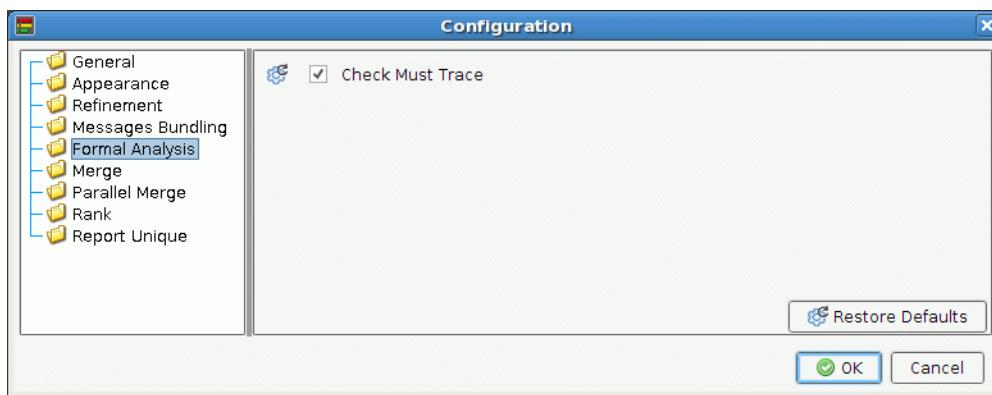
Note: These two options are related to the *Messages* dialog box that can be opened by clicking the envelope button at the bottom right.

1.11.5 Configuring Formal Analysis Options

To view and configure formal analysis related options, click *Formal Analysis* in the left pane in the *Configuration* dialog box.

[Figure 1-85 on page 84](#) displays the *Configuration* dialog box with *Formal Analysis* related options.

Figure 1-85 Configuration -- Formal Analysis Options



The *Formal Analysis* category has following options:

- **Check Must Trace** —This option indicates if the assertion properties must have a trace in order to be marked as *Proved*. By default, this option is enabled.
 - If enabled, the assertions must have a trace for the status to be marked as *Proved*.
 - If disabled, the assertion status can be marked as *Proved* without a trace.

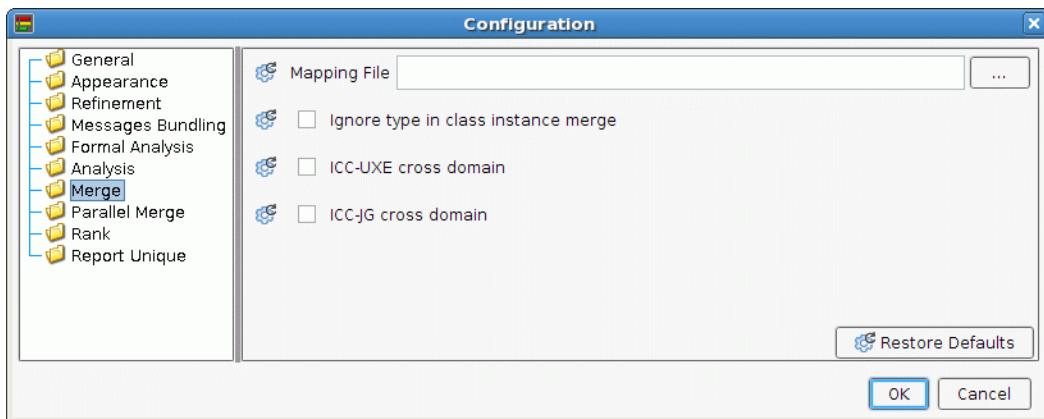
Note: This option impacts the *Formal Status Grade* calculation.

1.11.6 Configuring Merge Options

To view and configure merge related options, click *Merge* in the left pane in the *Configuration* dialog box.

[Figure 1-86 on page 85](#) displays the *Configuration* dialog box with *Merge* related options.

Figure 1-86 Configuration -- Merge Options



The *Merge* category has following options:

- **Mapping File**—To specify the entities mapping file for merge operations. This file may include one or more `merge_config` commands. For more details on the `merge_config` commands, see [Specifying Merge Configuration](#) on page 267. The file that you specify here is read and considered while performing the merge operation. If `merge_config` command is run from the command line and a mapping file is also specified here, then both are considered at the time of merge.

- **Ignore type in class instance merge**—To allow merging of class objects even if the type is different. By default, the name and the type of class instance is compared during the merge operation. If the name of the class instance is same but the type is different, then a merge conflict is reported and coverage is not merged for that class instance. By selecting this option, only the name of the class object is matched during the merge operation (the type is ignored). The CLI command to set this option is:

```
config merge.ignore_type_in_class_instance_merge -set true
```

- **ICC-UXE cross domain**—To merge ICC and UXE runs into the same hierarchy.

- **ICC-JG cross domain**—To merge ICC and JG runs into the same hierarchy.

For more details on cross domain merge, see `-cross_domain` option in [Merging Data](#) on page 259.

Note: The cross domain merge options can also be set using the `merge` command. There is an OR relation between the option set in the *Configuration* dialog box and the `merge` command. For more details on the `merge` command, see [Merging Data](#) on page 259.

1.11.7 Configuring Parallel Merge Options

Merging is a time-consuming process which involves significant computation and heavy disk access. In case you want to analyze many runs (thousands of runs), you might have to wait long before you can analyze the merged results.

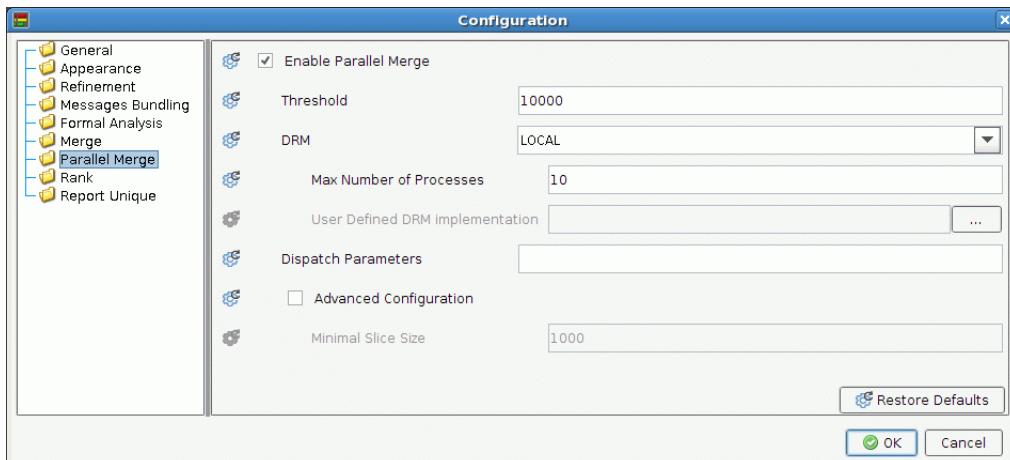
To save time, you can accelerate the merge calculation by utilizing multiple processes (potentially on multiple hosts). You can configure IMC such that the merge calculation is dispatched to multiple processes using the local host or to different hosts using DRM systems such as LSF and SGE. Such a merge arrangement is known as Parallel merge.

Note: Parallel merge is not always the best choice. There is an overhead associated with it, such as distributing the merge to multiple processes (process creation, communication, synchronization, and so on.) Parallel merge should not be used for small cases where regular incremental merge completes quickly.

To view and configure parallel merge related options, click *Parallel Merge* folder in the left pane in the *Configuration* dialog box.

[Figure 1-87 on page 86](#) displays the *Configuration* dialog box with *Parallel Merge* related options.

Figure 1-87 Configuration -- Parallel Merge Options



The *Parallel Merge* category has following options:

- *Enable Parallel Merge* check box—This option is not selected by default. Select this option to enable all other options in this page.
- *Threshold* field —This option lets you specify the number of runs after which the parallel merge should be performed. The default threshold value is 10000, which means that the

parallel merge will be performed only after the number of runs for merge exceeds the threshold value of 10000.

Note: The threshold value is for the number of runs. When merging multiple runs the runs are grouped by simulation engine. The merge is done for each group and the threshold is also evaluated for each group to merge. Therefore, there is a possibility that the number of total runs is greater than the threshold and still parallel merge is not used (because runs are from multiple engines).

- **DRM drop-down list** —This drop-down lets you specify the DRM type to be used for launching the processes. By default, Local (`LOCAL`) is used. You can select any of the following DRMs:
 - `LSF` — To use the integration with IBM Platform LSF for launching the processes.
 - `SGE` — To use open source batch-queuing system, Grid Engine for launching the processes.
 - `UGE` — To use Univa Grid Engine for launching the processes.
 - `OPENLAVA` — To use OpenLava workload scheduler for launching the processes.
 - `NC` — To use NetworkComputer based job scheduler for launching the processes.
 - `UGE` — To use Univa Grid Engine for launching the processes.
 - `LOCAL` — To schedule execution of runs in parallel or serial mode on the local machine for launching the processes.
 - `USER_DEFINED` — To use DRM that you created using the DRM API for launching the processes.
- **Max Number of Processes** field —This option lets you specify the maximum number of processes to be used while merging. The default value specified is 10. For Local DRM, the valid values are 2 to 50. For any other DRM, you can specify any integer 2 or more than 2.
- **User Defined DRM Implementation** field —If DRM is specified as `USER_DEFINED`, then *User Defined DRM Implementation* field becomes active. In this field, specify the TCL file (that includes the DRM API) to be used. For more details on creating a DRM API for parallel merge, see [Creating User-defined DRM API for Parallel Merge/Parallel Ranking/Parallel report_unique](#) on page 89.
- **Dispatch Parameters** field — This option lets you specify the dispatch parameters for the DRM. For more details on dispatch parameters, see the vManager User Guide.

- *Advanced Configuration* check box—This option is not selected by default. Select this option to enable specifying the number of runs to be sent to each process. After you select this check box, the *Minimal Slice Size* option becomes active.
- *Minimal Slice Size* field —This option lets you specify the number of runs to be sent to a single process. The default value is 1000. Valid values are 100 to 1000.

Note: To enable parallel merge in CLI mode, use:

```
config parallel_merge.enable_parallel_merge -set true
```

Note: While performing a merge, parallel merge also takes into consideration the settings specified with the `merge_config`.

Files Generated After Parallel Merge Process

After the parallel merge process, a log file named `processes_logs.log_<timestamp>` is generated in the current working directory. This log file summarizes all the process activities that took place during merge.

[Figure 1-88](#) on page 88 displays a sample of processes log file.

Figure 1-88 Log File Generated After Parallel Merge (Sample)

Merge Summary						
Configurations used:						
Threshold: 10000						
Number of process: 10						
DRM Type: DRM_CONTROL_LSF						
DRM Config dir: <DRM_config_dir>						
DRM Dispatch Parameters:						
Index	Start Time	End Time	Running Time	Pending Time	Number Of Runs	
1	Sun Aug 03 09:51:55 IDT 2014	Sun Aug 03 09:56:49 IDT 2014	00:04:53	00:00:27	1240	
2	Sun Aug 03 09:51:55 IDT 2014	Sun Aug 03 09:56:49 IDT 2014	00:04:53	00:00:26	1240	
3	Sun Aug 03 09:51:55 IDT 2014	Sun Aug 03 09:57:30 IDT 2014	00:05:34	00:00:26	1240	
4	Sun Aug 03 09:51:55 IDT 2014	Sun Aug 03 09:56:49 IDT 2014	00:04:53	00:00:25	1240	
5	Sun Aug 03 09:51:55 IDT 2014	Sun Aug 03 09:56:49 IDT 2014	00:04:53	00:00:24	1240	
6	Sun Aug 03 09:51:55 IDT 2014	Sun Aug 03 09:56:49 IDT 2014	00:04:53	00:00:24	1240	
7	Sun Aug 03 09:51:55 IDT 2014	Sun Aug 03 09:56:49 IDT 2014	00:04:53	00:00:23	1240	
8	Sun Aug 03 09:51:55 IDT 2014	Sun Aug 03 09:57:30 IDT 2014	00:05:34	00:00:22	1240	
9	Sun Aug 03 09:51:55 IDT 2014	Sun Aug 03 09:56:49 IDT 2014	00:04:53	00:00:21	1240	
10	Sun Aug 03 09:51:55 IDT 2014	Sun Aug 03 09:56:49 IDT 2014	00:04:53	00:00:20	1245	
12405 ucds were merged in: 00:06:03						

The above screen shows the summary of the merge. Below the summary (in the `processes_logs.log` file), a unified unordered sequential list of all the outputs coming from the running processes are shown.

In addition to the above `processes_logs.log` file, DRM reports are also generated in the current working directory under a directory named `merge_logs`.

1.11.7.1 Creating User-defined DRM API for Parallel Merge/Parallel Ranking/Parallel report_unique

To create a user-defined DRM API for parallel merge, parallel ranking or parallel report_unique, implement the following procedures in a TCL file:

- get_name
- is_drm_ready
- submit_job
- get_job_status
- terminate_job
- suspend_job
- resume_job
- get_job_exit_code
- submit_job_array (Optional Procedure)

Note: The TCL file you create is specified in the *User Defined DRM Implementation* field of the *Configuration* dialog box.

get_name

Returns the name of the DRM

Arguments: None

Example:

```
proc get_name {} {  
    return "my_lsf"  
}
```

is_drm_ready

Checks if DRM is set up correctly in the environment. Returns 1 if DRM is set up correctly, otherwise returns 0.

Arguments: None

Example:

```
proc is_drm_ready {} {  
    set out [run_command_return_output lsid]  
    return [regexp LSF $out]  
}
```

submit_job

Submits job to the DRM based on the given `job_template` argument; and returns the `job_id` of the submitted job.

Arguments: `job_template`

`job_template` is a TCL associative array that can contain following keys:

Key	Description	Note
command	Command to be run	
arguments	Command arguments	Might not exist
name	Name of the job	Might not exist
dispatch_params	DRM dispatch parameters	Might not exist
dir	Working directory of the job	Might not exist
output	Output redirection for the job	Might not exist
error	Error redirection for the job	Might not exist
env	Environment variable to set in job environment. It can be in the following format: VAR1=VALUE1 VAR2=VALUE2	Might not exist

Example:

```
proc submit_job {job_template} {  
    upvar $job_template jt
```

```
set args [join [get_bsub_args jt] ]
set env {}
if { [info exists jt(env)] } {
    set env $jt(env)
}
set out [run_command_return_output bsub $args $env]
return [get_job_id $out]
}
```

get_job_status

Queries the DRM and returns the status of the specified job. Status can be any of the following:

- UNKNOWN — Status is unknown
- WAITING — Job submitted to DRM
- SUSPENDED — Job was suspended
- RUNNING — Job is running on remote host
- COMPLETED — Job has completed running
- TERMINATED — Job was terminated (killed)

Arguments: job_id

Example:

```
proc get_job_status {job_id} {
    # run bjobs
    set bjobs_out [run_command_return_output bjobs $job_id]
    return [parse_bjobs_out $bjobs_out $job_id]
}
```

terminate_job

Terminates/kills the specified job.

Arguments: job_id of the job to terminate

Example:

```
proc terminate_job {job_id} {
    exec bkill $job_id
```

}

suspend_job

Suspends the specified job.

Arguments: `job_id` of the job to suspend

Note: If DRM does not support job suspension, then you can leave this procedure with empty implementation.

Example:

```
proc suspend_job {job_id} {
    exec bstop $job_id
}
```

resume_job

Resumes the specified job.

Arguments: `job_id` of the job to resume

Note: If DRM does not support job suspension, then you can leave this procedure with empty implementation.

Example:

```
proc resume_job {job_id} {
    exec bresume $job_id
}
```

get_job_exit_code

Returns the exit status of the process run by the job. It is undefined if the job has not completed yet.

Arguments: `job_id`

Example:

```
proc get_job_exit_code {job_id} {
    set bhistout [ run_command_return_output bhist "-l $job_id" ]
    if { [string first "Job <$job_id>" $bhistout] == -1 } {
        error "Unexpected output from bhist -l $job_id:\n$bhistout"
```

```
        }
        if { [string first "Done successfully" $bhistout] != -1 } {
            return 0;
        }
        regexp {Exited with exit code ([0-9]+).} $bhistout m c
        if { ! [info exists c] } {
            error "Unexpected output from bhist -l $job_id:\n$bhistout"
        }
        return $c
    }
}
```

submit_job_array (Optional Procedure)

This is an optional procedure that can be used to enable submitting multiple jobs together to improve performance. This procedure returns list of job ids.

Arguments:

- `job_template` (same as in [submit_job](#))
- `size`— Is number of jobs in the array

Note: When using job array, each job environment must include the `DRM_CONTROL_JOBINDEX_VAR` environment variable. It should include the name of another environment variable which would hold the index of the job in the array. For example, in LSF it would hold the `LSB_JOBINDEX` variable name.

1.11.7.2 User-Defined DRM Example

```
#####
# API procedures
#####

# get_name
# Return: the name of the DRM
proc get_name {} {
    return "my_lsf"
}

# is_drm_ready
# Check that the DRM is setup correctly in the environment
# Returns 1 if DRM is set correctly, 0 otherwise.
proc is_drm_ready {} {
```

```
set out [run_command_return_output lsid]
return [regexp LSF $out]
}

# submit_job
# Submits new job
# Parameters:
# job_template Associative array including job specification
# Return: The id of the submitted job
proc submit_job {job_template} {
    upvar $job_template jt
    set args [join [get_bsub_args jt] ]
    set env {}
    if { [info exists jt(env)] } {
        set env $jt(env)
    }
    set out [run_command_return_output bsub $args $env]
    return [get_job_id $out]
}

# get_job_status
# Query the DRM for the job status
# Parameters:
# job_id Id of the job
# Return: Job status [UNKNOWN, WAITING, SUSPENDED_WAITING, RUNNING,
# SUSPENDED_RUNNING, COMPLETED, TERMINATED]
proc get_job_status {job_id} {
    # run bjobs
    set bjobs_out [run_command_return_output bjobs $job_id]
    return [parse_bjobs_out $bjobs_out $job_id]
}

# terminate_job
# Kills the job
# Parameters:
# job_id Id of the job to terminate
proc terminate_job {job_id} {
    exec bkill $job_id
}
```

```
# suspend_job
# Suspends the job
# Parameters:
#   job_id Id of the job to suspend
proc suspend_job {job_id} {
    exec bstop $job_id
}

# resume_job
# Resumes the job
# Parameters:
#   job_id Id of the job to resume
proc resume_job {job_id} {
    exec bresume $job_id
}

# get_job_exit_code
# Returns the exit code of the job
# Parameters:
#   job_id Id of the job
# Return: The exit code of the job
proc get_job_exit_code {job_id} {
    set bhistout [ run_command_return_output bhist "-l $job_id" ]

    if { [string first "Job <$job_id>" $bhistout] == -1 } {
        error "Unexpected output from bhist -l $job_id:\n$bhistout"
    }

    if { [string first "Done successfully" $bhistout] != -1 } {
        return 0;
    }

    regexp {Exited with exit code ([0-9]+).} $bhistout m c
    if { ! [info exists c] } {
        error "Unexpected output from bhist -l $job_id:\n$bhistout"
    }
    return $c
}
```

```
#####
# Auxiliary procedures
#####

proc get_job_id {out} {
    regexp {Job <([0-9]+)> is submitted to.*} $out m s
    if { ! [info exists s] } {
        error "Unexpected output from bjobs: $out"
    }
    return $s
}

proc run_command_return_output { cmd {args ""} {env ""} } {
    set channel [ open "|/bin/env $env $cmd $args" r ]
    fconfigure $channel -buffering none
    return [ read $channel ]
}

proc get_bsub_args { job_template } {
    upvar $job_template jt;

    set result {};

    #set dispatch params
    if { [info exists jt(dispatch_params)] } {
        lappend result $jt(dispatch_params);
    }

    #set job name
    if { [info exists jt(name)] } {
        lappend result -j
        lappend result $jt(name)
    }

    #set job working dir
    if { [info exists jt(dir)] } {
        lappend result -cwd
        lappend result $jt(dir)
    }
}
```

```
#set output redirection
if { [info exists jt(output)] } {
lappend result -o
lappend result $jt(output)
}

#set error redirection
if { [info exists jt(error)] } {
lappend result -e
lappend result $jt(error)
}

#set the command to run
lappend result $jt(command);

#set the command arguments
if { [info exists jt(arguments)] } {
lappend result $jt(arguments);
}

return $result;
}

proc parse_bjobs_out {bjobs_out job_id} {
    set lines [split $bjobs_out \n]
    if { [llength $lines] < 2 } {
        return UNKNOWN
    }

    set cols [regexp -inline -all -- {\S+} [lindex $lines 1] ]
    if { [llength $cols] < 3 || [ string first [lindex $cols 0] $job_id] != 0 } {
        return UNKNOWN
    }

    return [lsf_status_to_drm_status [lindex $cols 2] ]
}

proc lsf_status_to_drm_status { lsf_status } {
    switch $lsf_status {
```

Incisive Metrics Center User Guide

```
PEND {
    return WAITING
}

PSUSP {
    return SUSPENDED
}

RUN {
    return RUNNING
}

USUSP {
    return SUSPENDED
}

SSUSP {
    return SUSPENDED
}

DONE {
    return COMPLETED
}

EXIT {
    return TERMINATED
}

UNKWN {
    return UNKNOWN
}

WAIT {
    return WAITING
}

ZOMBI {
    return TERMINATED
}
}

}
```

1.11.8 Configuring Parallel Ranking Options

Ranking is a time-consuming process which involves significant computation and heavy disk access. In case you want to analyze many runs (thousands of runs), you might have to wait long before you can analyze the ranked results.

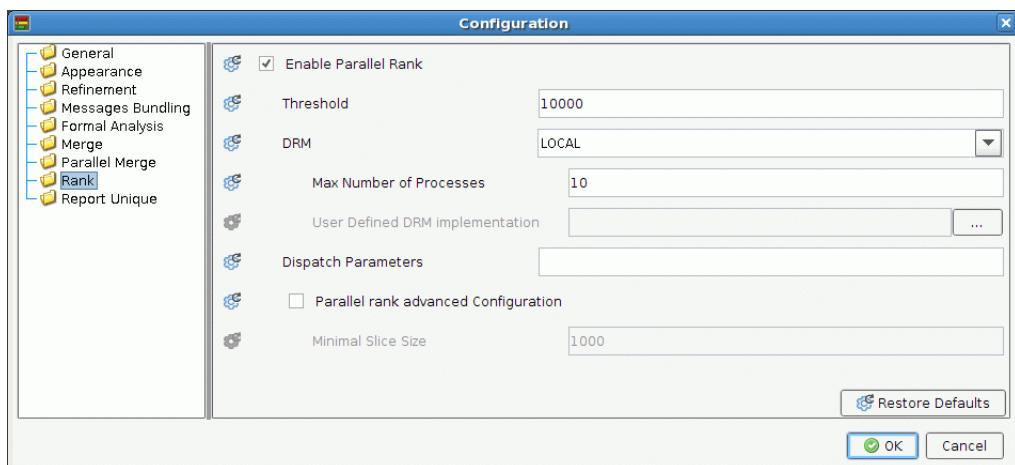
To save time, you can accelerate the ranking calculation by utilizing multiple processes (potentially on multiple hosts). You can configure IMC such that the ranking calculation is dispatched to multiple processes using the local host or to different hosts using DRM systems such as LSF and SGE.

Note: Parallel ranking is not always the best choice. There is an overhead associated with it, such as distributing the rank operation to multiple processes (process creation, communication, synchronization, and so on.) Parallel ranking should not be used for small cases.

To view and configure parallel ranking related options, click *Rank* folder in the left pane in the *Configuration* dialog box.

[Figure 1-89 on page 99](#) displays the *Configuration* dialog box with *Rank* related options.

Figure 1-89 Configuration -- Rank Options



The *Rank* category has following options:

- *Enable Parallel Rank* check box—This option is not selected by default. Select this option to enable all other options in this page.
- *Threshold* field—This option lets you specify the number of runs after which the parallel ranking should be performed. The default threshold value is 10000, which means that the

parallel ranking will be performed only after the number of runs for ranking exceeds the threshold value of 10000.

- **DRM drop-down list** —This drop-down lets you specify the DRM type to be used for launching the processes. By default, Local (`LOCAL`) is used. You can select any of the following DRMs:
 - `LSF` — To use the integration with IBM Platform LSF for launching the processes.
 - `SGE` — To use open source batch-queuing system, Grid Engine for launching the processes.
 - `UGE` — To use Univa Grid Engine for launching the processes.
 - `OPENLAVA` — To use OpenLava workload scheduler for launching the processes.
 - `NC` — To use NetworkComputer based job scheduler for launching the processes.
 - `LOCAL` — To schedule execution of runs in parallel or serial mode on the local machine for launching the processes.
 - `USER_DEFINED` — To use DRM that you created using the DRM API for launching the processes.
- ***Max Number of Processes* field** —This option lets you specify the maximum number of processes to be used while ranking. The default value specified is 10. For Local DRM, the valid values are 2 to 50. For any other DRM, you can specify any integer 2 or more than 2.
- ***User Defined DRM Implementation* field** —If DRM is specified as `USER_DEFINED`, then *User Defined DRM Implementation* field becomes active. In this field, specify the TCL file (that includes the DRM API) to be used. For more details on creating a DRM API for parallel ranking, see [Creating User-defined DRM API for Parallel Merge/Parallel Ranking/Parallel report_unique](#) on page 89.
- ***Dispatch Parameters* field** — This option lets you specify the dispatch parameters for the DRM. For more details on dispatch parameters, see the vManager User Guide.
- ***Advanced Configuration* check box**—This option is not selected by default. Select this option to enable specifying the number of runs to be sent to each process. After you select this check box, the *Minimal Slice Size* option becomes active.
- ***Minimal Slice Size* field** —This option lets you specify the number of runs to be sent to a single process. The default value is 1000. Valid values are 100 to 1000.

Note: To enable parallel ranking in CLI mode, use:

```
config rank.enable_parallel_rank -set true
```

Files Generated After Parallel Rank Process

After the parallel rank process, a log file named `rank_processes_logs.log_<timestamp>` is generated in the current working directory. This log file summarizes all the process activities that took place during rank.

[Figure 1-90 on page 101](#) displays a sample of processes log file.

Figure 1-90 Log File Generated After Parallel Rank (Sample)

Parallel Work Summary					
Configurations used:					
Threshold: 10000					
Number of process: 10					
DRM Type: DRM_CONTROL_LSF					
DRM Config dir: <DRM_Config_dir>					
DRM Dispatch Parameters:					
Index	Start Time	End Time	Running Time	Pending Time	Number Of Runs
1	Sun Aug 03 09:51:55 IDT 2014	Sun Aug 03 09:56:49 IDT 2014	00:04:53	00:00:27	1240
2	Sun Aug 03 09:51:55 IDT 2014	Sun Aug 03 09:56:49 IDT 2014	00:04:53	00:00:26	1240
3	Sun Aug 03 09:51:55 IDT 2014	Sun Aug 03 09:57:30 IDT 2014	00:05:34	00:00:26	1240
4	Sun Aug 03 09:51:55 IDT 2014	Sun Aug 03 09:56:49 IDT 2014	00:04:53	00:00:25	1240
5	Sun Aug 03 09:51:55 IDT 2014	Sun Aug 03 09:56:49 IDT 2014	00:04:53	00:00:24	1240
6	Sun Aug 03 09:51:55 IDT 2014	Sun Aug 03 09:56:49 IDT 2014	00:04:53	00:00:24	1240
7	Sun Aug 03 09:51:55 IDT 2014	Sun Aug 03 09:56:49 IDT 2014	00:04:53	00:00:23	1240
8	Sun Aug 03 09:51:55 IDT 2014	Sun Aug 03 09:57:30 IDT 2014	00:05:34	00:00:22	1240
9	Sun Aug 03 09:51:55 IDT 2014	Sun Aug 03 09:56:49 IDT 2014	00:04:53	00:00:21	1240
10	Sun Aug 03 09:51:55 IDT 2014	Sun Aug 03 09:56:49 IDT 2014	00:04:53	00:00:20	1245
12405 ucds were processed in: 00:06:03					

The above screen shows the summary of the rank process. Below the summary (in the `rank_processes_logs.log` file), a unified unordered sequential list of all the outputs coming from the running processes is shown.

1.11.9 Configuring Parallel Unique Coverage Report Options

The `report_unique` operation is a time-consuming process which involves significant computation and heavy disk access. In case you want to analyze many runs (thousands of runs), you might have to wait long before you can analyze the results.

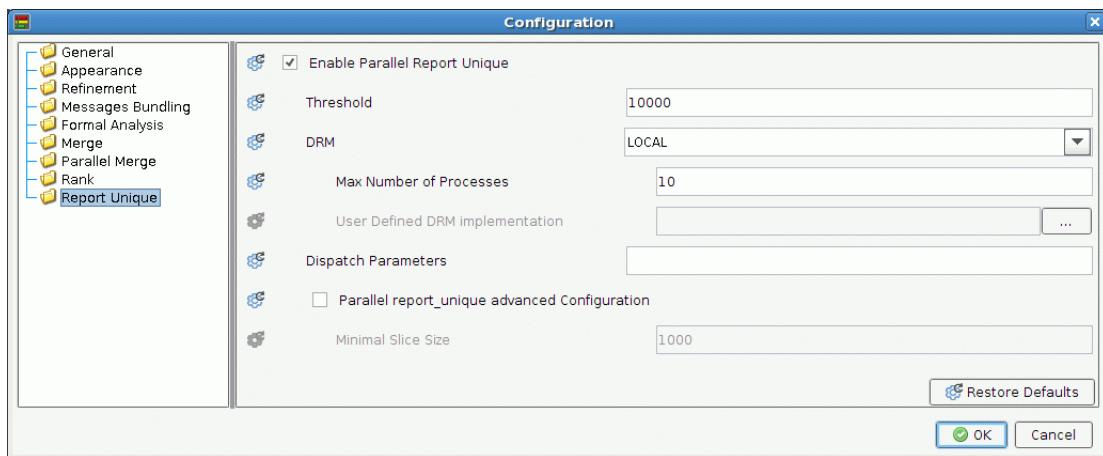
To save time, you can accelerate the performance of `report_unique` operation by utilizing multiple processes (potentially on multiple hosts).

Note: Parallel unique coverage report is not always the best choice. There is an overhead associated with it, such as distributing the `report_unique` operation to multiple processes (process creation, communication, synchronization, and so on.) Parallel unique coverage report should not be used for small cases. In addition, unique coverage report is not supported for expression and FSM metrics types.

To view and configure parallel report unique options, click *Report Unique* folder in the left pane in the *Configuration* dialog box.

[Figure 1-89](#) on page 99 displays the *Configuration* dialog box with *Parallel Report Unique* options.

Figure 1-91 Configuration -- Parallel Report Unique Options



The *Report Unique* category has following options:

- **Enable Parallel Report Unique** check box—This option is not selected by default. Select this option to perform `report_unique` operation using multiple processes. This will enable all other options in this page.
- **Threshold** field—This option lets you specify the number of runs after which the parallel report unique should be performed. The default threshold value is 10000, which means that the parallel report unique will be performed only after the number of runs for reporting exceeds the threshold value of 10000.
- **DRM** drop-down list—This drop-down lets you specify the DRM type to be used for launching the processes. By default, Local (`LOCAL`) is used. You can select any of the following DRMs:
 - **LSF**—To use the integration with IBM Platform LSF for launching the processes.

- SGE — To use open source batch-queuing system, Grid Engine for launching the processes.
 - UGE — To use Univa Grid Engine for launching the processes.
 - OPENLAVA — To use OpenLava workload scheduler for launching the processes.
 - NC — To use NetworkComputer based job scheduler for launching the processes.
 - LOCAL — To schedule execution of runs in parallel or serial mode on the local machine for launching the processes.
 - USER_DEFINED— To use DRM that you created using the DRM API for launching the processes.
- *Max Number of Processes* field —This option lets you specify the maximum number of processes to be used by parallel report unique operation. The default value specified is 10. For Local DRM, the valid values are 2 to 50. For any other DRM, you can specify any integer 2 or more than 2.
- *User Defined DRM Implementation* field —If DRM is specified as USER_DEFINED, then *User Defined DRM Implementation* field becomes active. In this field, specify the TCL file (that includes the DRM API) to be used. For more details on creating a DRM API for parallel report unique operation, see [Creating User-defined DRM API for Parallel Merge/Parallel Ranking/Parallel report_unique](#) on page 89.
- *Dispatch Parameters* field — This option lets you specify the dispatch parameters for the DRM. For more details on dispatch parameters, see the vManager User Guide.
- *Parallel report_unique Advanced Configuration* check box—This option is not selected by default. Select this option to enable specifying the number of runs to be sent to each process. After you select this check box, the *Minimal Slice Size* option becomes active.
- *Minimal Slice Size* field —This option lets you specify the number of runs to be sent to a single process. The default value is 1000. Valid values are 100 to 1000.

Note: To enable parallel report_unique in CLI mode, use:

```
config rank.enable_parallel_report_unique -set true
```

Incisive Metrics Center User Guide

Analyzing Metrics Data

After loading a run, you can analyze metrics data scored in that run. This chapter describes how to analyze the following metrics in the IMC GUI:

- [Code Coverage](#)
- [FSM Coverage](#)
- [Functional Coverage](#)

2.1 Code Coverage

Code coverage measures how thoroughly a testbench exercises the lines of HDL code that describe a design. The coverage results reveal areas of the design that have not been fully tested, or that did not meet desired coverage criteria.

Code coverage includes:

- [Block Coverage](#)
- [Expression Coverage](#)
- [Toggle Coverage](#)

2.1.1 Block Coverage

Block coverage identifies the lines of code that get executed during a simulation run. It helps you determine if the various testbenches exercise the statements in a block. You can analyze block coverage using the *Block* page of IMC GUI.

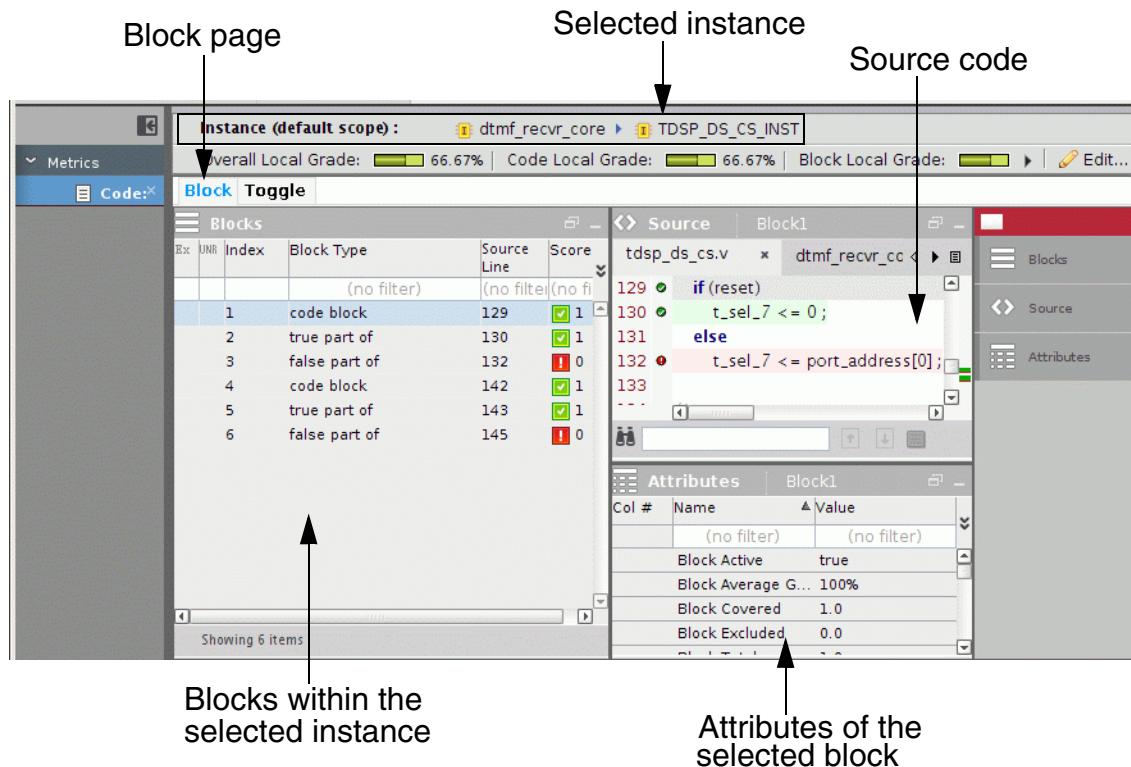
To launch the *Block* page:

1. On the *Metrics* page, select the instance or type for which you want to analyze block coverage.
2. Click the *Block* icon from the *Analyze* toolbar.

Note: Alternatively, you can right-click the instance or type and select *Block Analysis*.

The *Block* page is displayed in [Figure 2-1](#) on page 106.

Figure 2-1 Blocks Page



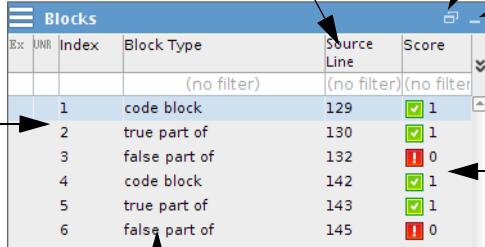
On the *Block* page, you can:

- [Identify Covered and Uncovered Blocks](#)
- [View Source Code](#)
- [View Attributes of Selected Block](#)
- [Export Table Data to a CSV File](#)
- [View Excluded, Covered, or Uncovered Items Only](#)
- [Generate HTML Block Coverage Report](#)

2.1.1.1 Identify Covered and Uncovered Blocks

The *Blocks* pane displays the blocks and branches of the selected instance or type. [Figure 2-2 on page 107](#) displays the *Blocks* pane.

Figure 2-2 Blocks Pane



The screenshot shows the 'Blocks' pane with a table of code blocks. The table has columns: Ex, UNR, Index, Block Type, Source Line, and Score. The 'Block Type' column contains labels like 'code block', 'true part of', 'false part of', etc. The 'Score' column contains values like 1, 0, or !0. The 'UNR' column contains icons. Row labels include 'no filter' and block IDs 1 through 6. The table is annotated with arrows pointing to specific features:

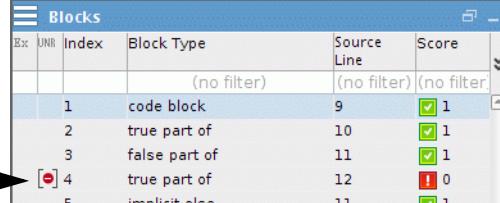
- Block ID**: Points to the 'Index' column.
- Line number**: Points to the 'Source Line' column.
- Type of block**: Points to the 'Block Type' column.
- Covered or Uncovered status**: Points to the 'Score' column, showing icons for checked (green), unchecked (red), and !0 (yellow).
- Tools & Options**: Points to a double-headed arrow icon in the top right of the table header.
- Toggle floating**: Points to a floating window icon in the top right of the pane.
- Hide**: Points to a hide icon in the top right of the pane.

By default, the *Blocks* pane displays the block identification number, block type, source line number, and the coverage status of the block. However, you can add more columns or remove columns from the table using the [Tools & Options](#) double arrow button.

You can detach the pane using the [Toggle floating](#) option. You can hide the pane using the [Toggle auto-hide](#) option. You can also [filter](#) data in the table to display only the required items, [sort](#) table data, and also [search](#) for items that meet specific search criteria.

If a particular block is marked unreachable (UNR) by Incisive Enterprise Verifier (IEV), then the *UNR* column shows an icon, as shown in [Figure 2-3 on page 107](#).

Figure 2-3 Blocks Marked Unreachable by IEV



The screenshot shows the 'Blocks' pane with a table of code blocks. The table has columns: Ex, UNR, Index, Block Type, Source Line, and Score. The 'UNR' column is highlighted. Row labels include 'no filter' and block IDs 1 through 5. The table is annotated with arrows pointing to specific features:

- Block marked unreachable**: Points to the 'UNR' column.

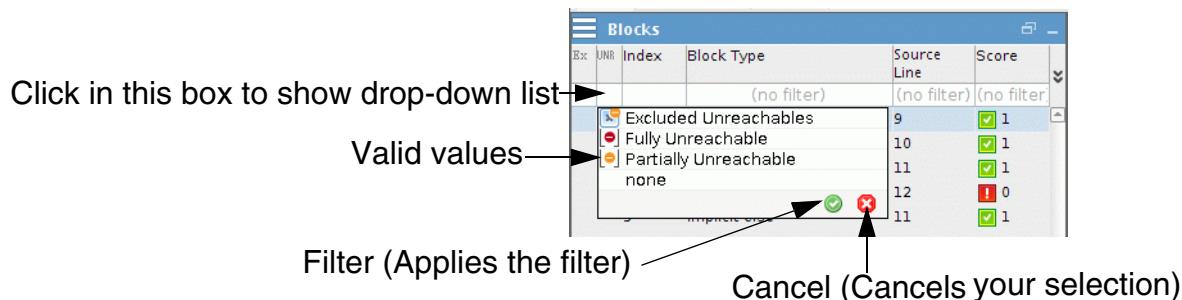
In the above table, Block 4 was identified and marked unreachable by IEV.

If you want to filter table data to show only the blocks that are marked unreachable by IEV, then:

1. Click in the Filter text box in the *UNR* column.

A drop-down is displayed with valid filter values, as shown in [Figure 2-4](#) on page 108.

Figure 2-4 Filtering Values in UNR Column



2. Select an appropriate value from the list. You can select any of the following values:

- Excluded Unreachables—To show only the items that were unreachable and were later marked excluded in IMC.
- Fully Unreachable—To show only the items that were marked fully unreachable by IEV.
- Partially Unreachable (not applicable for blocks)—To show only the items that were marked partially unreachable by IEV.
- None—To show the items that are not marked as either fully unreachable or partially unreachable.

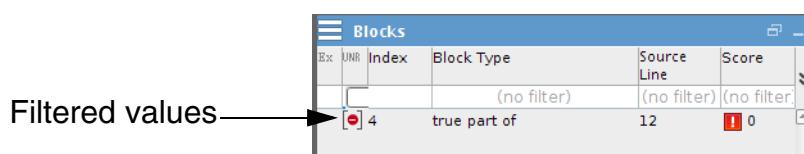
You can also select multiple values by holding the **Ctrl** key.

For example, to show only the items that were marked fully unreachable, select *Fully Unreachable* from the list.

3. Click the *Filter* icon to confirm the filtering action.

[Figure 2-5](#) on page 108 shows the filtered results.

Figure 2-5 Filtered Values in UNR Column



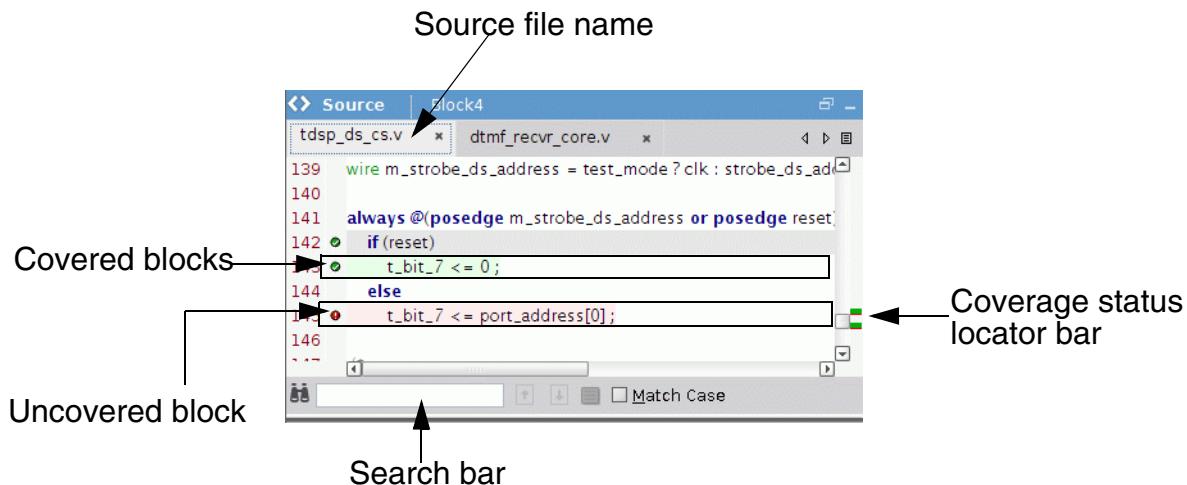
The Blocks pane now shows only the blocks that were marked fully unreachable by IEV.

For details on the UNR flow and how items are marked UNR by IEV, see the *IEV User Guide*.

2.1.1.2 View Source Code

When you click on a block in the *Blocks* pane, the corresponding source code is shown in the *Source* pane, as shown in [Figure 2-6](#) on page 109.

Figure 2-6 Source Pane



In the *Source* pane, a covered block is highlighted green and an uncovered block is highlighted red. If a source line includes both covered and uncovered blocks, then the source line is highlighted yellow.

Note: The same color coding is used in the coverage status locator bar. Using the coverage status locator bar, you can quickly reach the next covered or uncovered block. When you place the cursor on the coverage status locator bar, you can view the block ID and the coverage count for that block.

Using the search bar on the *Source* pane, you can quickly search for a specific text in the source code. See [Performing a Search on the Source Code](#) for details on how to perform a search on the source code.

2.1.1.3 View Attributes of Selected Block

When you select a block in the *Blocks* pane or the *Source* pane, its attributes are displayed in the *Attributes* pane. [Figure 2-7](#) on page 110 displays the attributes associated with block *Block4*.

Figure 2-7 Attributes Pane

The screenshot shows the Attributes pane for a block named "Block4". The pane is a table with three columns: "Col #", "Name", and "Value". The table contains the following data:

Col #	Name	Value
	(no filter)	(no filter)
3	Block Total Weighted Coverage	1.0
	Block Total Weights	1.0
	Block Type	code block
	Block Uncovered	0.0
	Code Average Grade	100%
	Code Covered	1.0

Annotations in the image:

- "Block name" points to the "Block4" label at the top of the pane.
- "Column #" points to the "Col #" column header.
- "Attribute's name" points to the "Name" column header.
- "Attribute's value" points to the "Value" column header.

You can right-click on an attribute and select any of the following:

- Add Columns—To add the selected column in the *Blocks* pane
- Remove Columns—To remove the selected column from the *Blocks* pane

Note: Alternatively, you can double-click on an attribute to add it to the *Blocks* pane, and then again double-click it to remove it from the *Blocks* pane.

- Unfilter Table—To release an already applied filter
- Unsort Table—To unsort the table data
- Copy Cell—To copy the data in the selected cell and paste it in any editor outside of IMC
- Copy Row—To copy the data in the selected row and paste it in any editor outside of IMC

The *Col #* column of the *Attributes* pane shows the column number of the corresponding attribute. For example, a value 3 in the *Col #* column indicates that this attribute appears at the third place in the *Blocks* pane. No value in the *Col #* column indicates that the corresponding attribute is not displayed in the *Blocks* pane.

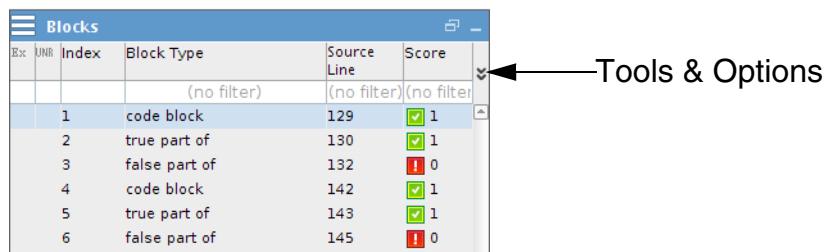
For a list of all of the attributes associated with a block, see [List of Attributes](#) on page 419.

2.1.1.4 Export Table Data to a CSV File

To export table data to a CSV file, perform the following steps:

1. Click the *Tools and Options* icon and select *Export to csv file*, as shown in [Figure 2-8](#) on page 111.

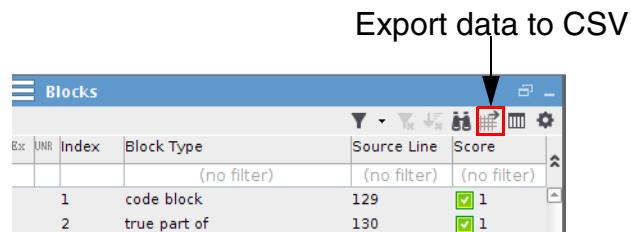
Figure 2-8 Export to CSV File



Ex	UNR	Index	Block Type	Source Line	Score
			(no filter)	(no filter)	(no filter)
1		1	code block	129	✓ 1
2		2	true part of	130	✓ 1
3		3	false part of	132	✗ 0
4		4	code block	142	✓ 1
5		5	true part of	143	✓ 1
6		6	false part of	145	✗ 0

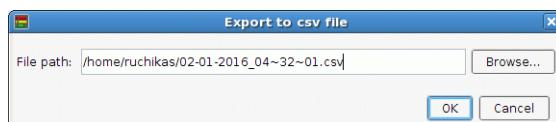
2. Select *Export data to csv file* from list of advanced buttons, as shown in [Figure 2-9](#) on page 111.

Figure 2-9 Export to CSV File



3. The *Export to csv file* dialog box appears, as shown in [Figure 2-10](#) on page 111.

Figure 2-10 Export to CSV File



4. In the *File path* field, you can specify the location and name of the file where the exported file must be saved. By default, it shows the location from which IMC was invoked and the file name is shown as <timestamp>.csv. You can change it, as required and click *OK*.

After you click *OK*, the table data is exported to the CSV file at the specified location.

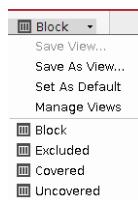
Note: The data included in the exported CSV file is based on the view selected at the time of generating the report.

2.1.1.5 View Excluded, Covered, or Uncovered Items Only

In the individual analysis pages, you can set a required view, and view only the excluded, covered, or uncovered items.

For example, [Figure 2-11](#) on page 112 shows the views available on the *Blocks* page.

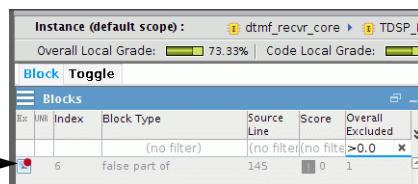
Figure 2-11 Views Available



You can select any of the following views:

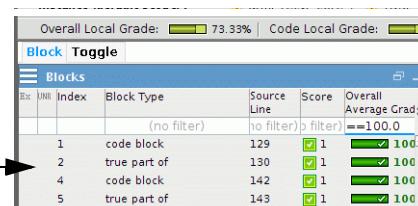
- Excluded—To show only the excluded items. This view adds an additional column, *Overall Excluded*, to the table; and filters the table data to show items with *Overall Excluded* > 0, as shown in [Figure 2-12](#) on page 112.

Figure 2-12 Excluded View

A screenshot of a table titled 'Blocks'. The columns are: Ex, UNR, Index, Block Type, Source Line, Score, and Overall Excluded. A filter bar at the top shows '(no filter)' for all columns except 'Overall Excluded', which is set to '>0.0'. A callout arrow points to the 'Overall Excluded' column with the text 'Shows only excluded items'.

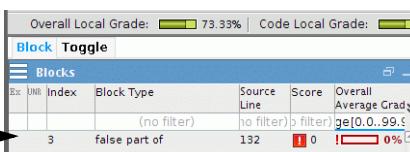
- Covered—To show only the covered items. This view adds an additional column, *Overall Average Grade*, to the table; and filters the table data to show items with *Overall Average Grade* == 100, as shown in [Figure 2-13](#) on page 112.

Figure 2-13 Covered View

A screenshot of a table titled 'Blocks'. The columns are: Ex, UNR, Index, Block Type, Source Line, Score, and Overall Average Grade. A filter bar at the top shows '(no filter)' for all columns except 'Overall Average Grade', which is set to '==100.0'. A callout arrow points to the 'Overall Average Grade' column with the text 'Shows only covered items'.

- Uncovered—To show only the uncovered items. This view adds an additional column, *Overall Average Grade*, to the table; and filters the table data to show items with *Overall Average Grade* in the range of 0.0 and 99.99, as shown in [Figure 2-14](#) on page 113.

Figure 2-14 Uncovered View



Index	Block Type	Source Line	Score	Overall Average Grade
3	false part of	132	0	0%

Note: These views (excluded, covered, and uncovered) are available in all analysis pages.

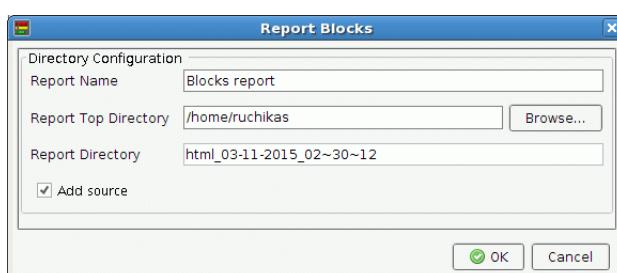
2.1.1.6 Generate HTML Block Coverage Report

To generate an HTML block coverage report:

1. Click the *Report* button in the Report toolbar.

The *Report Blocks* dialog box is displayed, as shown in [Figure 2-15](#) on page 113.

Figure 2-15 Report Blocks

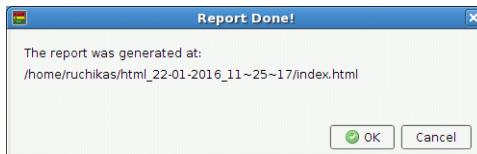


2. Specify the name of the report in the *Report Name* text box. By default, the name is specified as *Blocks Report*. This name will appear on the HTML report.
3. In the *Report Top Directory* field, specify the location where the report must be created. By default, it shows the location from which IMC was invoked.
4. Specify the name of the report directory in the *Report Directory* text box. By default, it shows the name as `html_<timestamp>`.
5. By default, the *Add source* check box is selected. This causes reporting of *Source Code* column in the report. If you do not want to include this column in the report, clear the *Add source* check box.

6. Click *OK* to generate the report.

The *Report Done* dialog box is displayed, as shown in [Figure 2-16](#) on page 114.

Figure 2-16 Report Done



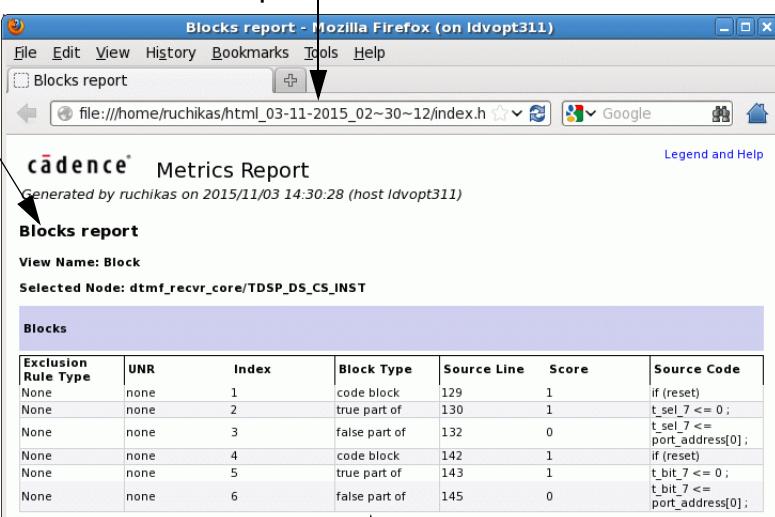
This dialog box shows the location where the report is generated.

7. Click *OK*.

You can navigate through the location shown above, and then open the `index.html` file in a Web browser.

[Figure 2-17](#) on page 114 shows the generated HTML report.

Figure 2-17 Blocks Report



Report location

Report name

List of blocks

Exclusion Rule Type	UNR	Index	Block Type	Source Line	Score	Source Code
None	none	1	code block	129	1	if (reset);
None	none	2	true part of	130	1	t_sel_7 <= 0;
None	none	3	false part of	132	0	t_sel_7 <= port_address[0];
None	none	4	code block	142	1	if (reset);
None	none	5	true part of	143	1	t_bit_7 <= 0;
None	none	6	false part of	145	0	t_bit_7 <= port_address[0];

The report shows the information such as, who generated the report and when the report was generated. The report lists the blocks in the selected instance or type.

The columns shown in the report, the sort order, and data shown in the report (filters) is based on the current state of the table at the time of generating the report. If required, you can adjust the columns that you want to include or remove from the table, apply sort order, and also apply filters before generating the report.

2.1.2 Expression Coverage

Expression coverage provides information on why a conditional piece of code was executed. It provides statistics for all expressions in the HDL code. You can analyze expression coverage using the *Expressions* page of the IMC GUI.

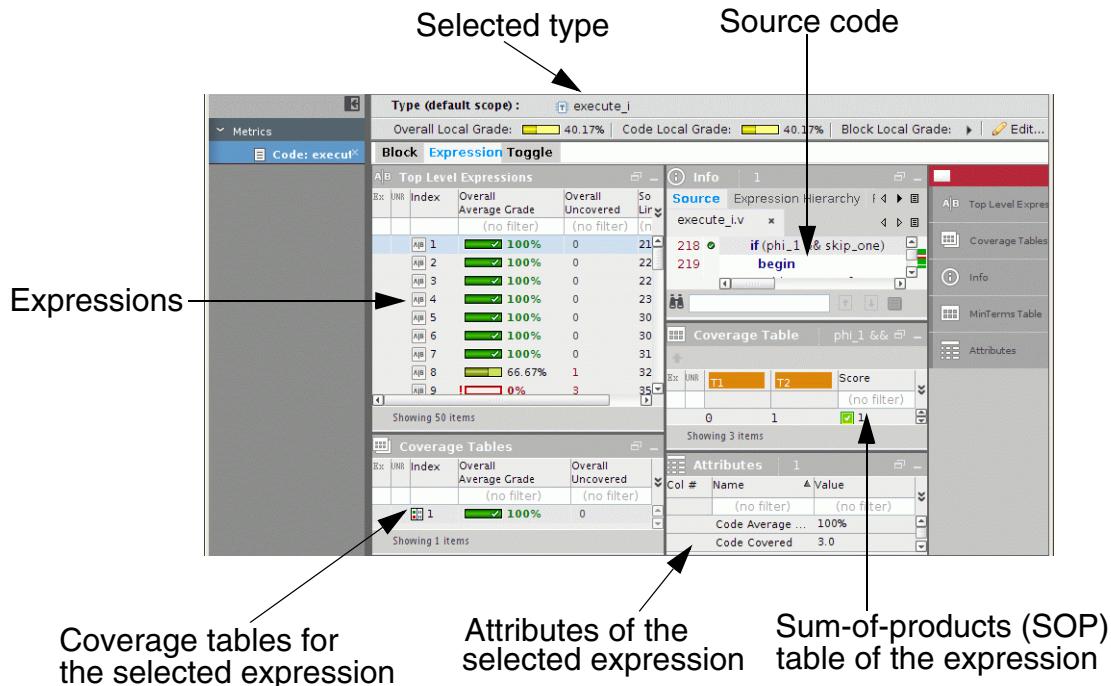
To launch the *Expressions* page:

1. On the *Metrics* page, select the instance or type for which you want to analyze expression coverage.
2. Click the *Expression* icon in the *Analyze* toolbar.

Note: Alternatively, you can right-click the instance or type and select *Expression Analysis*.

The *Expressions* page is displayed in [Figure 2-18](#) on page 115.

Figure 2-18 Expression Page



In the *Expression* page, you can:

- [Identify Covered and Uncovered Expressions](#)
- [View the SOP Table for Selected Expression](#)
- [View Source Code, Expression Hierarchy, and Expression](#)
- [View Attributes of Selected Expression](#)
- [Generate HTML Expression Coverage Report](#)

2.1.2.1 Identify Covered and Uncovered Expressions

The *Top-Level Expressions* pane displays the expressions within the selected instance or type. [Figure 2-19](#) on page 116 displays the *Top-Level Expressions* pane.

Figure 2-19 Top-Level Expressions Pane

Index	Overall Average Grade	Overall Uncovered	Source Line
1	100%	0	218
2	100%	0	222
3	100%	0	228
4	100%	0	232
5	100%	0	305
6	100%	0	309
7	100%	0	318
8	66.67%	1	324
9	0%	3	354

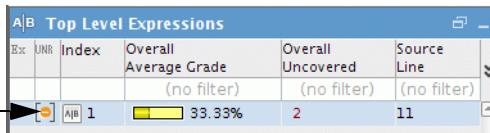
By default, the *Top-Level Expressions* pane displays the expression identification number, the overall average grade, number of uncovered terms, and the source line number. However, you can add more columns or remove columns from the table using the [Tools & Options](#) double arrow button. You can detach the pane using the [Toggle floating](#) option. You can hide the pane using the [Toggle auto-hide](#) option.

You can also [filter](#) data in the table to display only the required items, [sort](#) table data, and also [search](#) for items that meet specific search criteria. You can also filter the table data to show only excluded, covered, or uncovered items. For more details, see [View Excluded, Covered, or Uncovered Items Only](#) on page 112.

You can also export the table data to a CSV file, if required. For more details, see [Export Table Data to a CSV File](#) on page 110.

Note: If a particular expression is marked unreachable (UNR) by Incisive Enterprise Verifier (IEV), then the *UNR* column shows an icon, as shown in [Figure 2-20](#) on page 117.

Figure 2-20 Expressions Marked Unreachable by IEV



The screenshot shows a table titled "Top Level Expressions" with the following columns: Ex, UNR, Index, Overall Average Grade, Overall Uncovered, and Source Line. The "Index" column for Expression 1 contains a red icon with a minus sign, indicating it is marked as unreachable. The "Overall Average Grade" cell for Expression 1 shows "(no filter)" with a yellow bar at 33.33%. The "Overall Uncovered" cell for Expression 1 shows "2" with a red bar. The "Source Line" cell for Expression 1 shows "11" with a blue bar. The table has a header row with icons for sorting and filtering.

Top Level Expressions					
Ex	UNR	Index	Overall Average Grade	Overall Uncovered	Source Line
		Expression marked unreachable →	(no filter)	2	11

In the above table, Expression 1 was identified and marked unreachable by IEV.

Note: The color of the icon and the tooltip indicates if the expression was marked as fully unreachable or partially unreachable. A red color icon indicates fully unreachable. A fully unreachable icon shows the tooltip as **Fully Unreachable**. An orange color icon indicates partially unreachable. A partially unreachable icon shows the tooltip as **Partially Unreachable**.

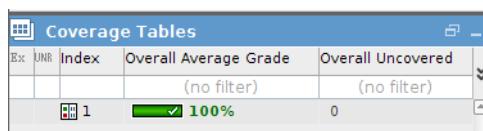
You can also filter table data to show only the items that are marked fully unreachable or partially unreachable by IEV. For more details, see [filtering UNR items](#).

For details on the UNR flow and how items are marked UNR by IEV, see the *IEV User Guide*.

2.1.2.2 View the SOP Table for Selected Expression

When you select an expression in the *Top-Level Expressions* pane, the tables associated with that expression are displayed in the *Coverage Tables* pane. [Figure 2-21](#) on page 117 displays tables associated with expression #1.

Figure 2-21 Coverage Tables Pane

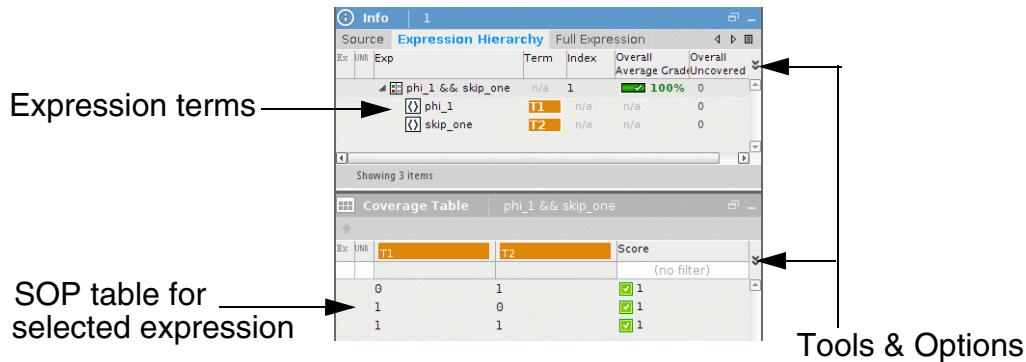


The screenshot shows a table titled "Coverage Tables" with the following columns: Ex, UNR, Index, Overall Average Grade, and Overall Uncovered. The "Index" column for Expression 1 contains a green icon with a checkmark, indicating it is marked as partially unreachable. The "Overall Average Grade" cell for Expression 1 shows "100%" with a green bar. The "Overall Uncovered" cell for Expression 1 shows "0". The table has a header row with icons for sorting and filtering.

Coverage Tables				
Ex	UNR	Index	Overall Average Grade	Overall Uncovered
		Expression 1	100%	0

In the given example, only one table is associated with the selected expression. When you click an expression in the *Coverage Tables* pane, the hierarchy tree for that expression is shown in the *Expression Hierarchy* page, and the SOP table is shown in the *Coverage Table* pane, as shown in [Figure 2-22](#) on page 118.

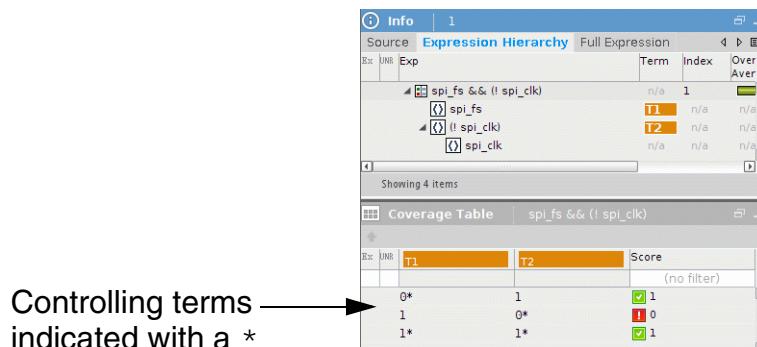
Figure 2-22 Expression Hierarchy and SOP Table



By default, Coverage Table shows the term numbers, such as, T1, T2, and so on. If you want to show the expression term from the source code instead of T1, T2, select the *Full expression in header column* option in the *Configuration* dialog box. For more details, see [Configuring Appearance Options](#) on page 80.

By default, there is no indication to show which expression terms are controlling terms. If you want to know which expression terms are controlling terms, select the *Control scoring expression indication* option in the *Configuration* dialog box. After you select this option, a * notation appears next to the controlling terms in expression tables in GUI, HTML reports, and ASCII reports, as shown in [Figure 2-23](#) on page 118. For more details, see [Configuring Appearance Options](#) on page 80.

Figure 2-23 Controlling Terms Have a * Notation



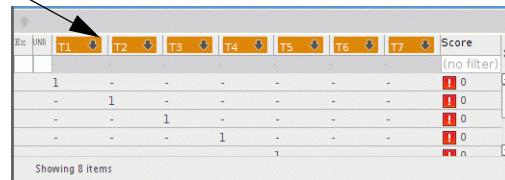
Note: For complex expressions, the *Coverage Table* pane shows an additional column named *Output*, and the pane in those cases is named as *Coverage Table with Output*.

Move Between Coverage Tables

If an expression has related sub-expressions, then the table headers with expression terms (for example T1, T2 and so on) will have a down-arrow to allow jumping to the related sub-expression table, as in [Figure 2-24](#) on page 119.

Figure 2-24 Move Between Coverage Tables (Go to sub-expression)

Click to go to next table
(T8/T9/T10)

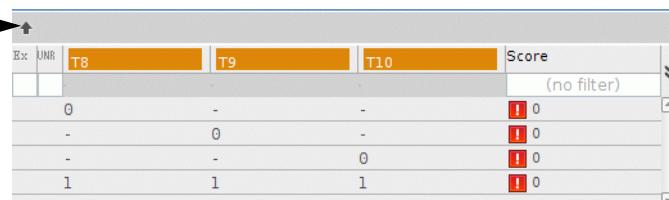


Ex	UNR	T1	T2	T3	T4	T5	T6	T7	Score
									(no filter)
1	-	-	-	-	-	-	-	-	! 0
-	1	-	-	-	-	-	-	-	! 0
-	-	1	-	-	-	-	-	-	! 0
-	-	-	1	-	-	-	-	-	! 0
-	-	-	-	1	-	-	-	-	! 0
-	-	-	-	-	1	-	-	-	! 0

After you click the down-arrow that appears in the column header, you are taken to the sub-expression table of that term, as shown in [Figure 2-25](#) on page 119.

Figure 2-25 Move Between Coverage Tables (Go to parent)

Click to go back to the parent



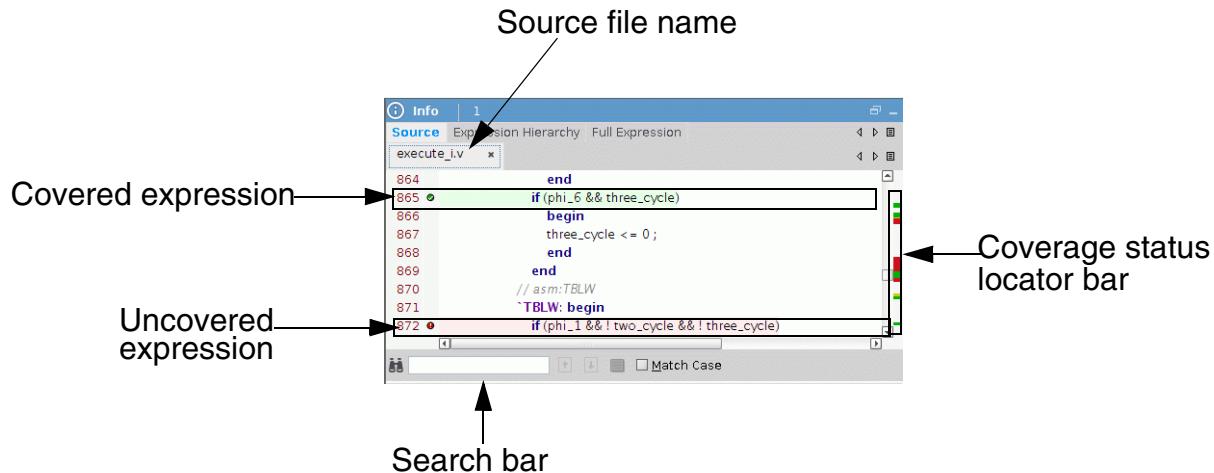
Ex	UNR	T8	T9	T10	Score
					(no filter)
0	-	-	-	-	! 0
-	0	-	-	-	! 0
-	-	-	0	-	! 0
1	1	1	1	1	! 0

You can click the up-arrow button and go back to the parent table.

2.1.2.3 View Source Code, Expression Hierarchy, and Expression

When you select an expression in the *Top-Level Expressions* pane, the corresponding source code is shown in the *Source* page of the *List tabs* pane, as shown in [Figure 2-26](#) on page 120.

Figure 2-26 Source Page



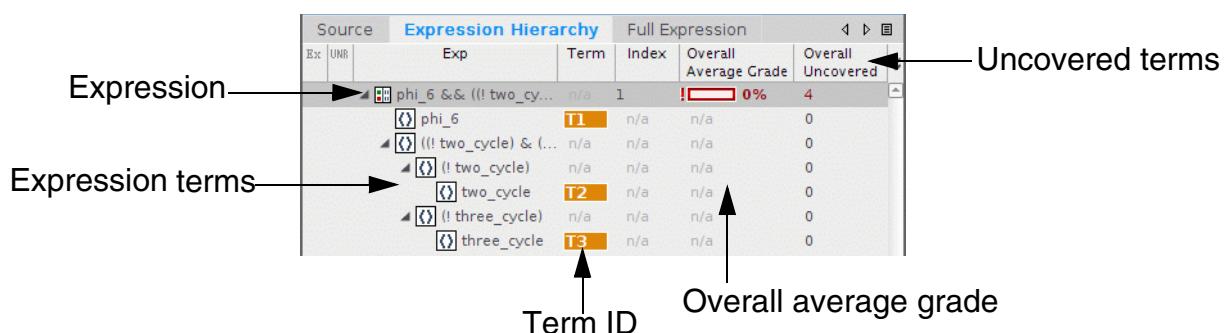
In the *Source* tab page, a covered expression is highlighted green and an uncovered expression is highlighted red.

Note: The same color coding is used in the coverage status locator bar. Using the coverage status locator bar, you can quickly reach the next covered or uncovered expression. When you place the cursor on the coverage status locator bar, you can view the expression ID and the grade for that expression.

Using the search bar on the *Source* tab page, you can quickly search for a specific text in the source code. See [Performing a Search on the Source Code](#) for details on how to perform a search on the source code.

To view the expression hierarchy tree for the selected expression, click the *Expression Hierarchy* tab. Figure 2-27 on page 120 shows a expression hierarchy tree.

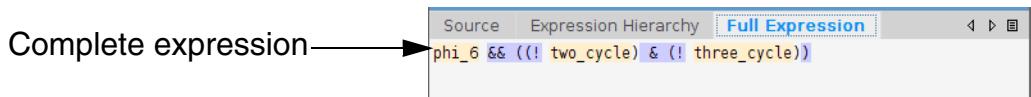
Figure 2-27 Expression Hierarchy Page



In the *Expression Hierarchy* page, the expression hierarchy tree of the selected expression is shown. All of the expression terms are listed as a tree structure.

At times, the expression is too large to display clearly on the *Expression Hierarchy* page. To view the complete expression, click the *Full Expression* tab, as shown in [Figure 2-28](#) on page 121.

Figure 2-28 Full Expression Page



2.1.2.4 View Attributes of Selected Expression

When you select an expression in the *Top-Level Expressions* pane or the *Source* tab page, its attributes are displayed in the *Attributes* pane. [Figure 2-29](#) on page 121 displays the attributes associated with expression 1.

Figure 2-29 Attributes Pane

A screenshot of the 'Attributes' pane for expression 1. The pane shows a table with columns for 'Col #' (Index), 'Name', and 'Value'. The table contains the following data:

Col #	Name	Value
1	(no filter)	(no filter)
2	Expression Total Weighted Co...	0.0
3	Expression Total Weights	3.0
4	Expression Uncovered	3.0
5	Index	29
6	Is Active	true
7	Is Vplan	false
8	Name	
9	Overall Average Grade	0%

You can right-click on an attribute and select any of the following:

- Add Columns—To add the selected column in the *Top-Level Expressions* pane
- Remove Columns—To remove the selected column from the *Top-Level Expressions* pane

Note: Alternatively, you can double-click on an attribute to add it to the *Top-Level Expressions* pane, and then again double-click it to remove it from the *Top-Level Expressions* pane.

- Unfilter Table—To release an already applied filter

- Unsort Table—To unsort the table data
- Copy Cell—To copy the data in the selected cell and paste it in any editor outside of IMC
- Copy Row—To copy the data in the selected row and paste it in any editor outside of IMC

The *Col #* column of the *Attributes* pane shows the column number of the corresponding attribute. For example, a value *0* in the *Col #* column for the attribute *Index* indicates that this column appears at the first place in the *Top-Level Expressions* pane. No value in the *Col #* column indicates that the corresponding attribute is not displayed in the *Top-Level Expressions* pane.

For a list of attributes associated with a expression, see [List of Attributes](#) on page 419.

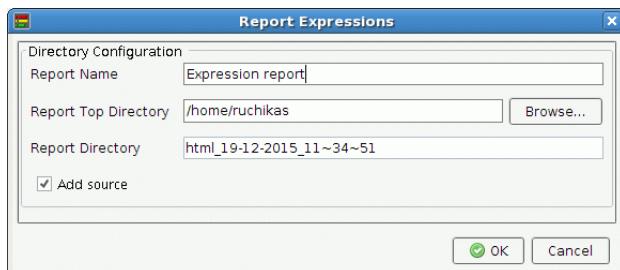
2.1.2.5 Generate HTML Expression Coverage Report

To generate an HTML expression coverage report:

1. Click the *Report* button in the Report toolbar.

The *Report Expressions* dialog box is displayed, as shown in [Figure 2-30](#) on page 122.

Figure 2-30 Report Expressions

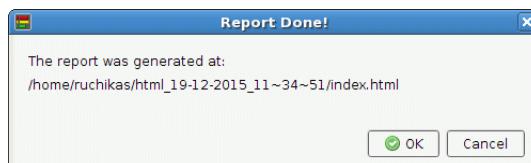


2. Specify the name of the report in the *Report Name* text box. By default, the name is specified as *Expressions Report*. This name will appear on the HTML report.
3. In the *Report Top Directory* field, specify the location where the report must be created. By default, it shows the location from which IMC was invoked.
4. Specify the name of the report directory in the *Report Directory* text box. By default, it shows the name as `html_<timestamp>`.
5. By default, the *Add source* check box is selected. This causes reporting of *Source Code* column in the report. If you do not want to include this column in the report, clear the *Add source* check box.

6. Click *OK* to generate the report.

The *Report Done* dialog box is displayed, as shown in [Figure 2-31](#) on page 123.

Figure 2-31 Report Done



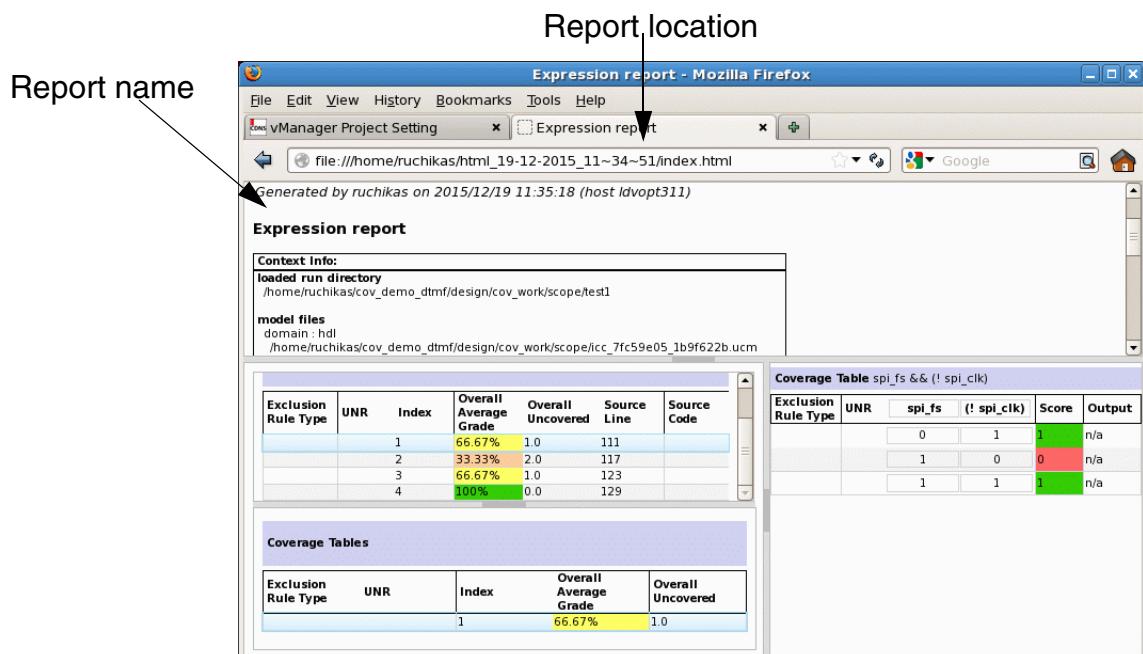
This dialog box shows the location where the report is generated.

7. Click *OK*.

You can navigate through the location shown above, and then open the `index.html` file in a Web browser.

[Figure 2-32](#) on page 123 shows the generated HTML report.

Figure 2-32 Expression Report



Exclusion Rule Type	UNR	Index	Overall Average Grade	Overall Uncovered	Source Line	Source Code
1			66.67%	1.0	111	
2			33.33%	2.0	117	
3			66.67%	1.0	123	
4			100%	0.0	129	

Exclusion Rule Type	UNR	spi_fs	(! spi_clk)	Score	Output
	0	1	1	n/a	
	1	0	0	n/a	
	1	1	1	n/a	

The report shows the information such as, who generated the report and when the report was generated. The report lists the expressions in the selected instance or type.

The columns shown in the report, the sort order, and data shown in the report (filters) is based on the current state of the table at the time of generating the report. If required, you can adjust the columns that you want to include or remove from the table, apply sort order, and also apply filters before generating the report.

2.1.3 Toggle Coverage

Toggle coverage provides information about the change of signals and ports, during a simulation run. It measures activity in the design, such as unused signals, signals that remain constant, or signals that have too few value changes. You can analyze toggle coverage using the *Toggle* page of the IMC GUI.

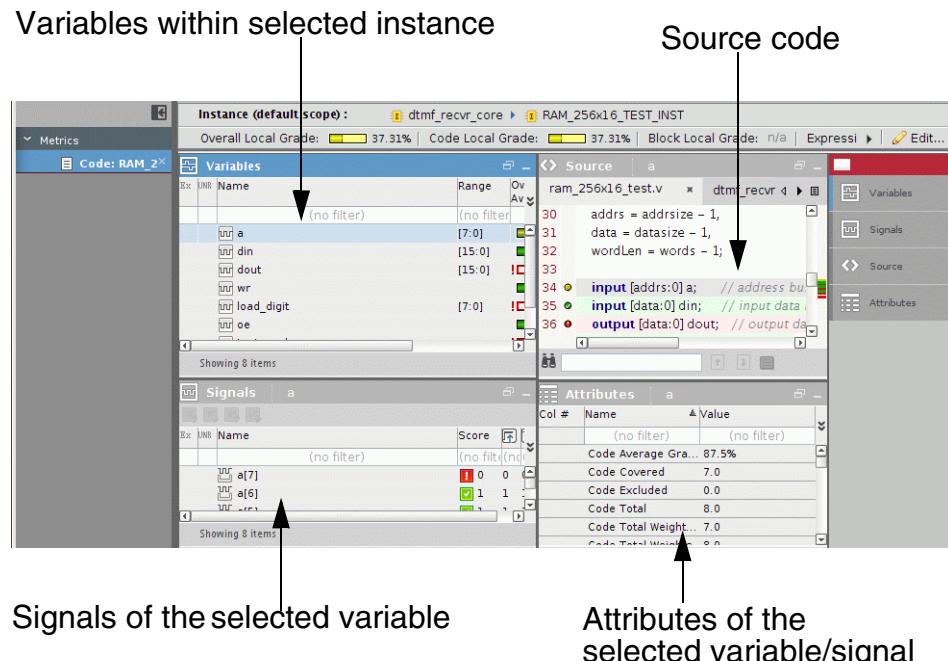
To launch the *Toggle* page from the *Metrics* page:

1. Select the instance for which you want to analyze toggle coverage.
2. Click the *Toggle* icon in the *Analyze* toolbar.

Note: Alternatively, you can right-click the instance or type and select *Toggle Analysis*.

The *Toggle* page is displayed in [Figure 2-33](#) on page 124.

Figure 2-33 Toggle Page



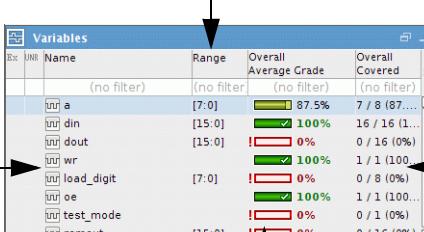
In the *Toggle* page, you can:

- [Identify Covered and Uncovered Signals and Variables](#)
- [View Source Code](#)
- [View Attributes of Selected Signal/Variable](#)
- [Generate HTML Toggle Coverage Report](#)

2.1.3.1 Identify Covered and Uncovered Signals and Variables

The *Variables* pane displays the covered and uncovered variables for the selected instance or type in the loaded run, as shown in [Figure 2-34](#) on page 125.

Figure 2-34 Variables Pane



The screenshot shows a table titled "Variables" with the following data:

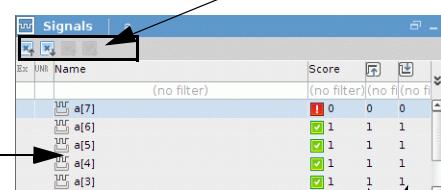
Ex	INR	Name	Range	Overall Average Grade	Overall Covered
		(no filter)	(no filter)	(no filter)	(no filter)
		u[7:0]	87.5%	7 / 8 (87.5%)	
		u[15:0]	100%	16 / 16 (100%)	
		u[15:0]	0%	0 / 16 (0%)	
		u[7:0]	100%	1 / 1 (100%)	
		u[7:0]	0%	0 / 8 (0%)	
		u[15:0]	100%	1 / 1 (100%)	
		u[15:0]	0%	0 / 1 (0%)	
		u[15:0]	0%	0 / 16 (0%)	

Annotations on the screenshot:

- "Variable range" points to the "Range" column.
- "Tools & Options" points to the double arrow icon in the top right corner.
- "List of variables" points to the list of variable names.
- "Overall covered grade" points to the "Overall Covered" column.
- "Overall average grade" points to the "Overall Average Grade" column.

The signals associated with the selected variable are displayed in the *Signals* pane, as shown in [Figure 2-35](#) on page 125.

Figure 2-35 Signals Pane



The screenshot shows a table titled "Signals" with the following data:

Ex	INR	Name	Score
		(no filter)	(no filter)
		a[7]	0 0 0
		a[6]	1 1 1
		a[5]	1 1 1
		a[4]	1 1 1
		a[3]	1 1 1

Annotations on the screenshot:

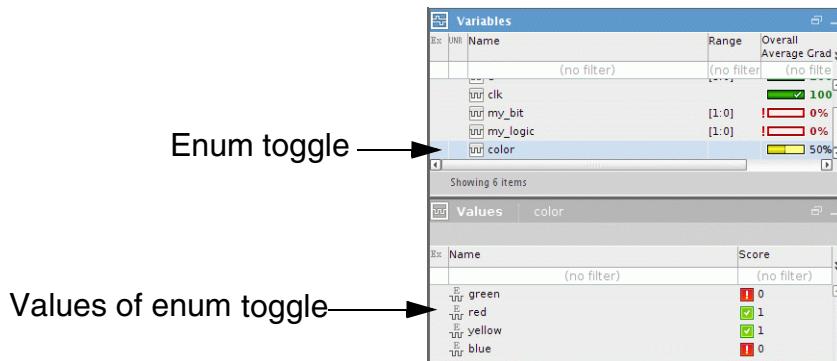
- "Signals associated with variable a" points to the list of signals: a[7], a[6], a[5], a[4], a[3].
- "Icons to apply exclusions" points to the double arrow icon in the top right corner.
- "Signal rise and fall counts" points to the last three columns of the table.

You can add more columns or remove columns from the table in the *Variables* pane and the *Signals* pane using the [Tools & Options](#) double arrow. You can also [filter](#) data in the table to display only the required items, [sort](#) table data, and also [search](#) for items that meet specific

search criteria. You can also filter the table data to show only excluded, covered, or uncovered items. For more details, see [View Excluded, Covered, or Uncovered Items Only](#) on page 112. You can also export the table data to a CSV file, if required. For more details, see [Export Table Data to a CSV File](#) on page 110.

Note: If you select an enumerated toggle in the *Variables* pane, then the signals pane shows the values of the enumerated toggle, as shown in [Figure 2-36](#) on page 126.

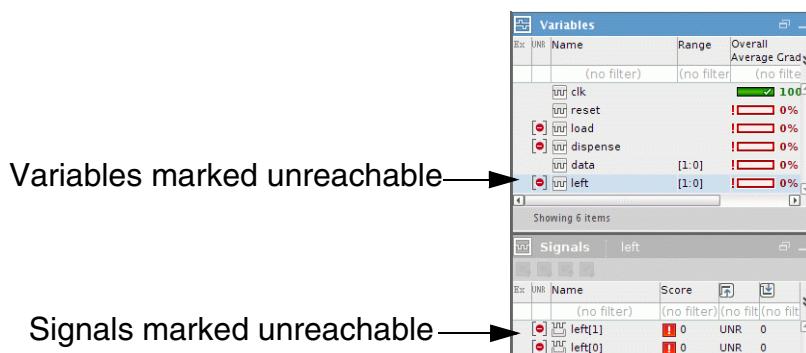
Figure 2-36 Signals Pane in the case of Enumerated Toggle



In the case of enum toggles, the rise and fall values are not applicable and therefore are not shown. The *Score* column shows the covered and uncovered status of the enumerated value.

If a particular signal is marked unreachable (UNR) by Incisive Enterprise Verifier (IEV), then the *UNR* column shows an icon, as shown in [Figure 2-37](#) on page 126.

Figure 2-37 Signals/Variables Marked Unreachable by IEV



A toggle variable is marked as:

- Fully Unreachable if all of its signals are marked as UNR.

- Partially Unreachable if at least one but not all of its signals are marked as UNR.

A toggle signal is marked as:

- Fully Unreachable if one or both (rise and fall) transitions are marked as UNR.

Note: Partially Unreachable marker is not used in case of toggle signals.

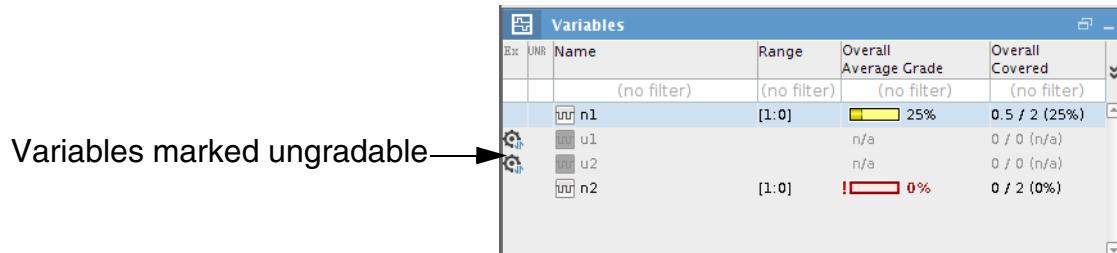
The color of the icon and the tooltip indicates if the signal was marked as fully unreachable or partially unreachable. A red color icon indicates fully unreachable. A fully unreachable icon shows the tooltip as Fully Unreachable. An orange color icon indicates partially unreachable. A partially unreachable icon shows the tooltip as Partially Unreachable.

You can also filter table data to show only the items that are marked fully unreachable or partially unreachable by IEV.

For details on the UNR flow and how items are marked UNR by IEV, see the *IEV User Guide*.

If a particular toggle variable is marked ungradable at the time of simulation run, then the *Ex* column shows ungradable icon for those variables, as shown in [Figure 2-38 on page 127](#).

Figure 2-38 Variables Marked Ungradable

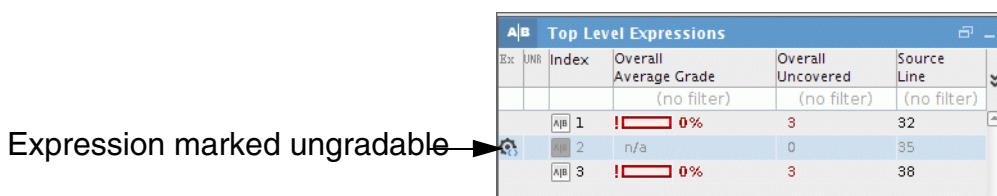


Ex	UNR	Name	Range	Overall Average Grade	Overall Covered
		(no filter)	(no filter)	(no filter)	(no filter)
		n1	[1:0]	25%	0.5 / 2 (25%)
		u1		n/a	0 / 0 (n/a)
		u2		n/a	0 / 0 (n/a)
		n2	[1:0]	0%	0 / 2 (0%)

You can also filter table data to show only the items that are marked ungradable.

If a particular expression is marked ungradable at the time of simulation run, then the *Ex* column shows ungradable icon for those variables, as shown in [Figure 2-39 on page 127](#).

Figure 2-39 Expressions Marked Ungradable



Ex	UNR	Index	Overall Average Grade	Overall Uncovered	Source Line
			(no filter)	(no filter)	(no filter)
		1	0%	3	32
		2	n/a	0	35
		3	0%	3	38

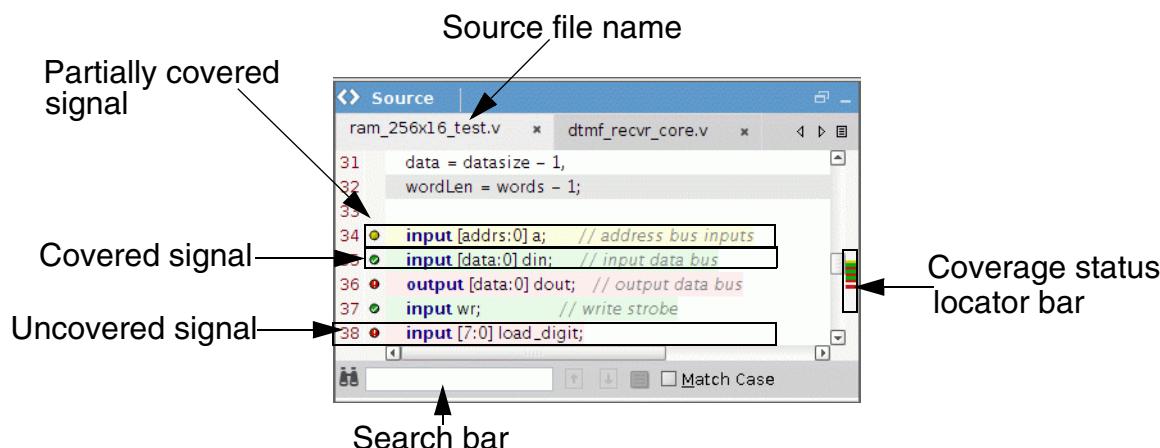
You can also filter table data to show only the items that are marked ungradable.

For details on items marked ungradable, see the *ICC User Guide*.

2.1.3.2 View Source Code

When you click on a variable or a signal in the *Variables* pane or the *Signals* pane, the corresponding source code is shown in the *Source* pane, as shown in [Figure 2-40](#) on page 128.

Figure 2-40 Source Pane



In the *Source* pane, a covered variable/signal is highlighted green, an uncovered variable/signal is highlighted red, and a partially covered signal/variable is highlighted yellow.

Note: The same color coding is used in the coverage status locator bar. Using the coverage status locator bar, you can quickly reach the next covered or uncovered signal. When you place the cursor on the coverage status locator bar, you can view the signal name and the overall grade for that signal.

Using the search bar on the *Source* pane, you can quickly search for a specific text in the source code.

2.1.3.3 View Attributes of Selected Signal/Variable

When you select a variable or signal, its attributes are displayed in the *Attributes* pane. [Figure 2-41](#) on page 129 displays the attributes associated with variable *a*.

Figure 2-41 Attributes Pane

Col #	Name	Value
	(no filter)	(no filter)
2	Name	false
	Is Vplan	false
	Kind	input
3	Overall All Excluded	0.0
4	Overall Average Grade	87.5%
5	Overall Covered	7.0
	Overall Excluded	0.0

You can right-click on an attribute and select any of the following:

- Add Columns—To add the selected column in the *Variables* pane
- Remove Columns—To remove the selected column from the *Variables* pane

Note: Alternatively, you can double-click on an attribute to add it to the *Variables* pane, and then again double-click it to remove it from the *Variables* pane.

- Unfilter Table—To release an already applied filter
- Unsort Table—To unsort the table data
- Copy Cell—To copy the data in the selected cell and paste it in any editor outside of IMC
- Copy Row—To copy the data in the selected row and paste it in any editor outside of IMC

The *Col #* column of the *Attributes* pane shows the column number of the corresponding attribute. For example, a value *0* in the *Col #* column for the attribute *Name* indicates that this column appears at the first place in the *Variables* pane. No value in the *Col #* column indicates that the corresponding attribute is not displayed in the *Variables* pane.

For a list of all of the attributes associated with a variable or a signal, see [List of Attributes](#) on page 419.

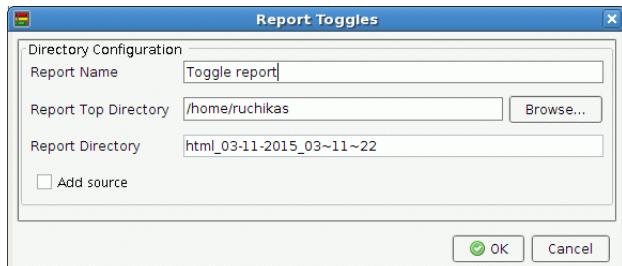
2.1.3.4 Generate HTML Toggle Coverage Report

To generate an HTML toggle coverage report:

1. Click the *Report* button in the Report toolbar.

The *Report Toggles* dialog box is displayed, as shown in [Figure 2-42](#) on page 130.

Figure 2-42 Report Toggles



2. Specify the name of the report in the *Report Name* text box. By default, the name is specified as *Toggle Report*. This name will appear on the HTML report.
3. In the *Report Top Directory* field, specify the location where the report must be created. By default, it shows the location from which IMC was invoked.
4. Specify the name of the report directory in the *Report Directory* text box. By default, it shows the name as `html_<timestamp>`.
5. By default, the *Add source* check box is selected. This causes reporting of *Source Code* column in the report. If you do not want to include this column in the report, clear the *Add source* check box.
6. Click *OK* to generate the report.

The *Report Done* dialog box is displayed, as shown in [Figure 2-43](#) on page 130.

Figure 2-43 Report Done



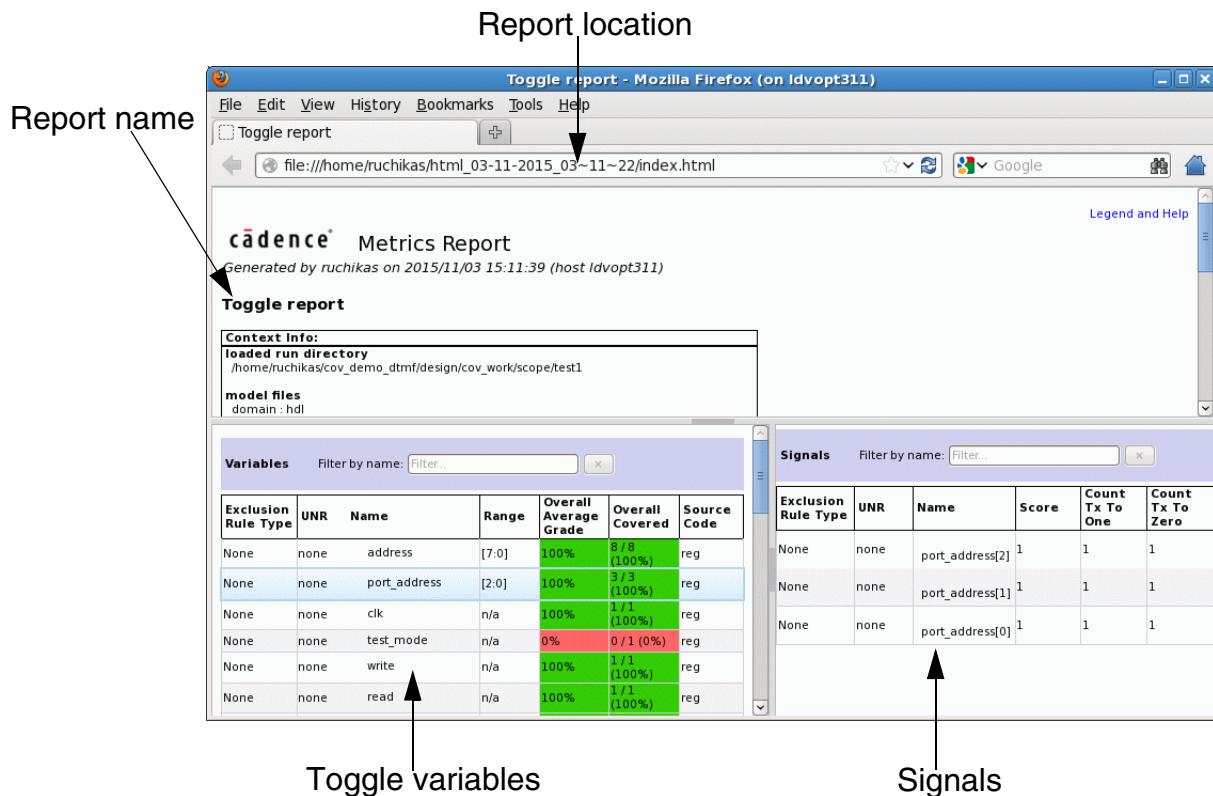
This dialog box shows the location where the report is generated.

7. Click *OK*.

You can navigate through the location shown above, and then open the `index.html` file in a Web browser.

[Figure 2-44](#) on page 131 shows the generated HTML report.

Figure 2-44 Toggle Report



The report shows the information such as, who generated the report and when the report was generated. It also includes the context information, such as coverage database location, model file (.ucm) and data file (.ucd) used to generate the report, the location and contents of the coverage configuration file, and the list of refinement files applied before generating the report.

The report lists the toggle information of the selected instance or type. It shows following tables:

- **Variables**—This table lists the toggle variables within the selected instance or type.
- **Signals**—This table lists the signals within the selected toggle variable.

The columns shown in the report, the sort order, and data shown in the report (filters) is based on the current state of the table at the time of generating the report. If required, you can adjust the columns that you want to include or remove from the table, apply sort order, and also apply filters before generating the report.

Note: After you have generated the report, you can filter data based on the *Name* field using the *Filter by name* text box provided at the top of the generated HTML report.

2.2 FSM Coverage

FSM coverage interprets the synthesis semantics of the HDL design and monitors the coverage of the FSM representation of control logic blocks in the design. It answers the question “Did I reach all of the states and cover all possible transitions or arcs in a given state machine?”

To analyze FSM coverage in IMC:

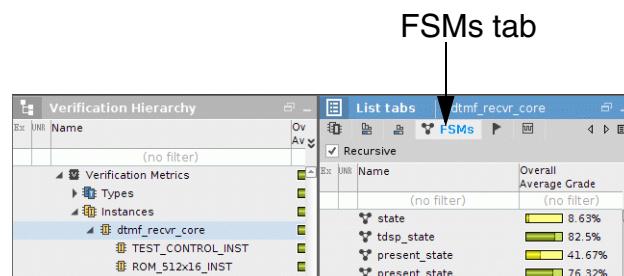
1. Navigate through the design hierarchy on the *Metrics* page and identify overall coverage of different state machines in the loaded run.
2. Launch the *FSM* page to perform a detailed analysis of different state machines.

2.2.1 Identify State Machine for FSM Coverage Analysis

To list all the state machines in the design for FSM analysis:

1. Click a top-level instance in the verification hierarchy tree.
2. Select *FSMs* tab in the right-hand pane and check the *Recursive* check box, as shown in [Figure 2-45 on page 132](#).

Figure 2-45 Metrics Page



The *FSMs* tab page displays the list of FSMs along with the overall covered and uncovered grade of that FSM. From this list, you can identify the FSM that you want to analyze in detail and then improve its coverage. Similarly, you can select a specific instance in the verification hierarchy tree and list the FSMs in that instance.

2.2.2 Launch the FSM Page

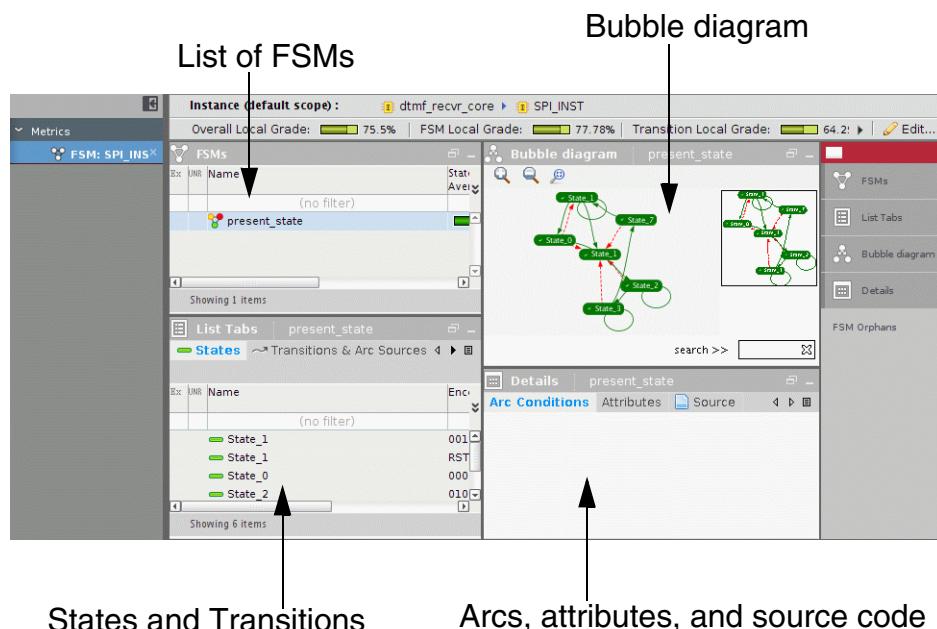
After identifying the FSM for further analysis, launch the *FSM* page. To launch the *FSM* page:

1. In the hierarchy tree, select the instance or type for which you want to analyze FSM coverage.
2. Click the *FSM* icon in the *Analyze* toolbar.

Note: Alternatively, you can right-click the instance or type and select *FSM Analysis*.

The *FSM* page is launched, as shown in [Figure 2-46](#) on page 133.

Figure 2-46 *FSM* Page



On the *FSM* page, you can:

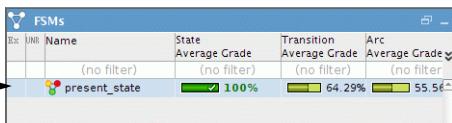
- [View the List of FSMs](#)
- [View Covered and Uncovered States and Transitions](#)
- [View the Bubble Diagram](#)
- [View the Underlying Source Code, Arc Conditions, and Attributes](#)
- [Generate HTML FSM Coverage Report](#)

Note: The *Locality* toolbar allows you to change the roll-up type. *Local* is the default, which shows state registers in the selected instance. When you click *Local*, the button is named as *Recursive*, which allows you to display state registers within all the children of the selected instance. If the button shows disabled, it indicates that you cannot change the roll-up type of the selected item.

2.2.2.1 View the List of FSMs

The *FSMs* pane shown in [Figure 2-47](#) on page 134 displays the list of FSMs in the selected instance or type.

Figure 2-47 **FSM Pane**



FSM state register →

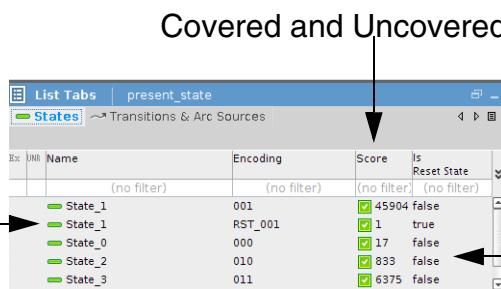
Tools & Options →

The *FSMs* pane, by default, shows the name of FSM and the overall grade (state, transition, and arc coverage). You can add more columns or remove columns from the table, as required. You can also filter data in the table to display only the required items, sort table data, and also search for items that meet specific search criteria.

2.2.2.2 View Covered and Uncovered States and Transitions

When you select a state register in the *FSMs* pane, the corresponding states are displayed in the *States* tab page, as shown in [Figure 2-48](#) on page 134.

Figure 2-48 **States**



Covered and Uncovered status →

State name →

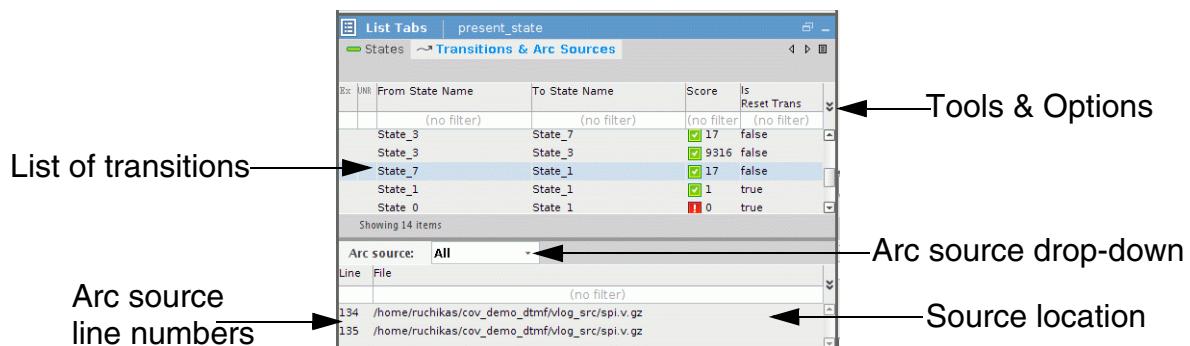
Reset state →

Tools & Options →

By default, the *States* tab page lists the different states of the selected state register, displays the encoding value of different states, coverage status and score (green for covered and red for uncovered states), and if the state is a reset state.

To view the transitions and arcs for the selected state register, click the *Transitions & Arc Sources* tab. [Figure 2-49](#) on page 135 displays the *Transitions & Arc Sources* tab page.

Figure 2-49 Transitions & Arc Sources



The *Transitions & Arc Sources* tab page displays all of the transitions that occurred in the selected state register and the coverage status for the transitions (green for covered and red for uncovered transition).

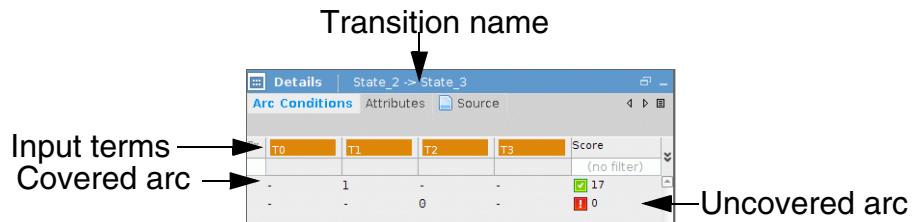
In addition, when you select the *Transitions & Arc Sources* tab, a separate pane is displayed to show the arc source line numbers and the source location for the selected transition. The *Arc Sources* drop-down has the following options:

- All—Lists all the arcs (from state, to state, and check related source lines) for the selected transition, and also highlights all the arcs in the *Source* tab page.
- Checks—Lists only the check related source line, and highlights the source line on which the check conditions are located in the *Source* tab page.
- From States—Lists only the from state related source lines of the selected transition, and highlights the from state arc in the *Source* tab page.
- To States —Lists only the to state related source lines of the selected transition, and highlights the to state arc in the *Source* tab page.

You can add more columns or remove columns from the tables shown on the *States* and *Transitions* page, as required. You can detach the pane using the *Toggle floating* option. You can hide the pane using the *Toggle auto-hide* option. You can also filter data in the table to display only the required items, sort table data, and also search for items that meet specific search criteria. You can also filter the table data to show only excluded, covered, or uncovered items. For more details, see [View Excluded, Covered, or Uncovered Items Only](#) on page 112. You can also export the table data to a CSV file, if required. For more details, see [Export Table Data to a CSV File](#) on page 110.

When you select a particular transition on the *Transitions and Arc Sources* page, the arcs corresponding to that transition are displayed in the *Arc Conditions* page, as shown in [Figure 2-50](#) on page 136.

Figure 2-50 Details Pane—Arc Conditions Tab

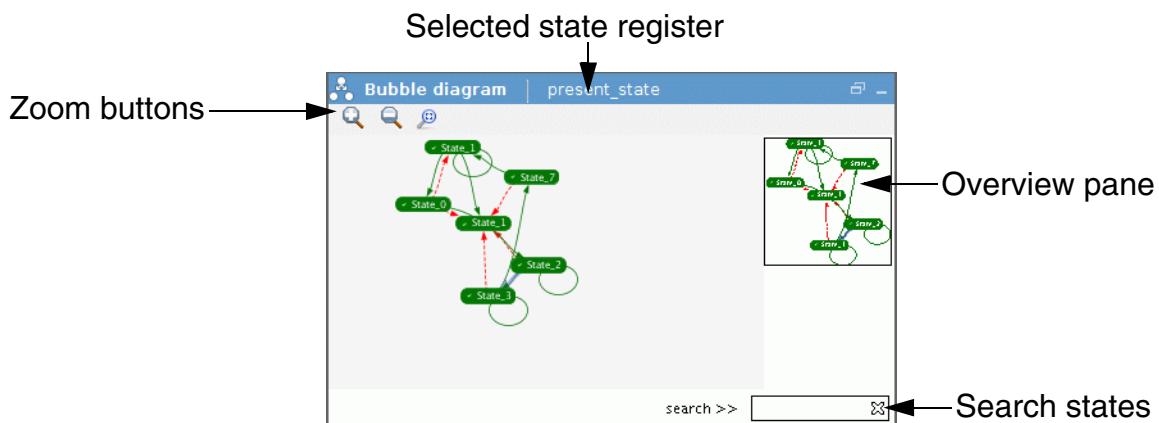


Notice that transition `State_2 -> State_3` happens under two input conditions, `-1--` and `--0-`, respectively.

2.2.2.3 View the Bubble Diagram

The *Bubble Diagram* pane of the *FSM* page displays a pictorial representation of the FSM state register. [Figure 2-51](#) on page 136 displays the bubble diagram of state register `present_state`.

Figure 2-51 Bubble Diagram



The coloring mechanism for states and transitions in the bubble diagram is as follows:

- Green—Indicates fully covered state or transition.
- Red—Indicates uncovered state or transition.

In the *Bubble Diagram* pane, you can:

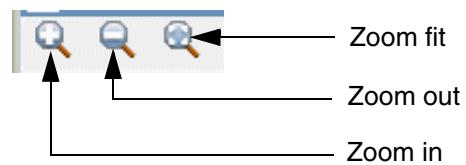
- [Change the Zoom-Level of the Bubble Diagram](#)
- [View Coverage Counts](#)

- Search Specific States

Change the Zoom-Level of the Bubble Diagram

Figure 2-52 on page 137 displays the Zoom buttons available to change the zoom-level of the bubble diagram.

Figure 2-52 The Zoom Buttons



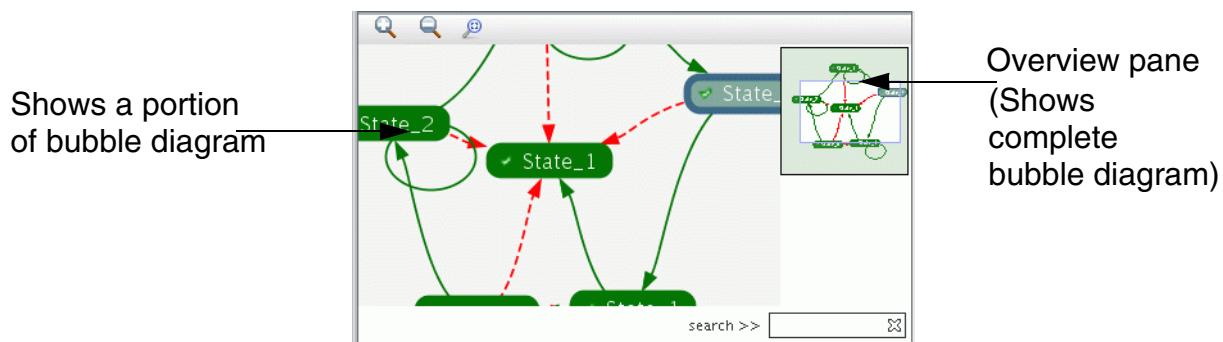
To change the zoom-level of a bubble diagram, click the:

- *Zoom out* button to reduce the size of bubble diagram.
- *Zoom fit* button to scale the bubble diagram to fit in the current size of the display area.
- *Zoom in* button to enlarge the bubble diagram.

Note: In addition to using the zoom buttons, you can also change the zoom-level by dragging the mouse with the right mouse-button depressed.

As you keep enlarging the bubble diagram, the bubble diagram might not be visible in the current width of the pane. However, the *Overview* pane continues to show the complete diagram, as shown in Figure 2-53 on page 137.

Figure 2-53 The Overview Pane



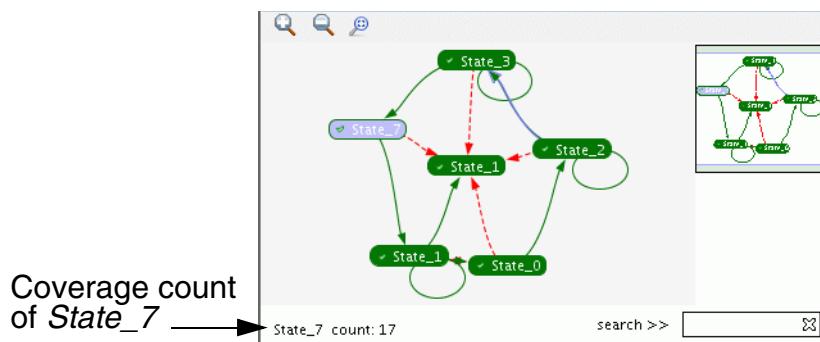
The *Overview* pane always shows the complete bubble diagram. The rectangle shown within the *Overview* pane indicates the current visible portion in the display area of the *Bubble*

Diagram pane. You can drag the rectangle shown in the *Overview* pane to control the visibility of the bubble diagram in the display area of the *Bubble Diagram* pane.

View Coverage Counts

To view the coverage count for a state or transition, move the cursor over a state or transition in the bubble diagram. The coverage count for that state or transition is displayed in the status bar in the lower-left hand corner, as shown in [Figure 2-54](#) on page 138.

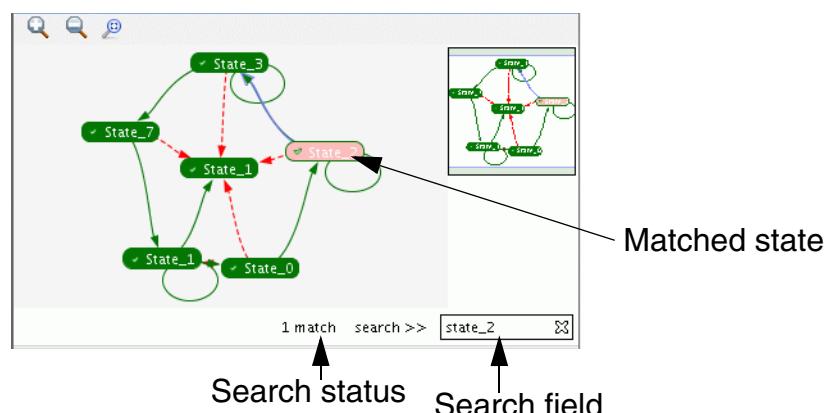
Figure 2-54 The Status Bar—Coverage Count



Search Specific States

To search for a specific state, type the name of the state in the search field, as shown in [Figure 2-55](#) on page 138.

Figure 2-55 The Status Bar—Search Field



Note: The state specified in the search field text box is highlighted in the bubble diagram.

2.2.2.4 View the Underlying Source Code, Arc Conditions, and Attributes

The *Details* pane of the *FSM* page shows additional information, such as the source code, attributes, and arcs of the selected state or transition. The *Details* pane has the following tabs:

- [Arc Conditions](#)
- [Attributes](#)
- [Source](#)

Arc Conditions

The *Arc Conditions* tab page displays all possible input conditions under which the transition took place. When you select a particular transition on the *Transitions and Arc Sources* page, the arcs corresponding to that transition are displayed in the *Arc Conditions* page, as shown in [Figure 2-56](#) on page 139.

Figure 2-56 Details Pane—Arc Conditions Tab

Transition name

Input terms

Covered arc

Uncovered arc

	T0	T1	T2	T3	Score
-	-	1	-	-	(no filter)
-	-	0	-	-	17

Uncovered arc

Notice that transition `State_2 -> State_3` happens under two input conditions, `-1--` and `--0-`, respectively.

Note: A transition is evaluated covered if any of the arcs for that transition are covered. A transition is evaluated uncovered if all of the arcs for that transition are uncovered. In the above report, transition `State_2 -> State_3` is considered covered because one of the arcs for this transition is covered.

Attributes

To view the attributes such as the file location, line number, overall grade, reset count, status, encoding value, and more information about the selected state or transition, click the *Attributes* tab. [Figure 2-57](#) on page 140 displays the attributes associated with state `State_1`.

Figure 2-57 Details Pane—Attributes Tab

Col #	Name	Value
	(no filter)	(no filter)
2	From State Line	122
	From State Name	State_2
5	Index	12
	Is Active	true
	Is Reset Trans	false
	Is Vplan	false
	Name	State_2 -> State_3

You can right-click on an attribute and select any of the following:

- Add Columns—To add the selected column in the *FSMs* pane
- Remove Columns—To remove the selected column from the *FSMs* pane

Note: Alternatively, you can double-click on an attribute to add it to the *FSMs* pane, and then again double-click it to remove it from the *FSMs* pane.

- Unfilter Table—To release an already applied filter
- Unsort Table—To unsort the table data
- Copy Cell—To copy the data in the selected cell and paste it in any editor outside of IMC
- Copy Row—To copy the data in the selected row and paste it in any editor outside of IMC

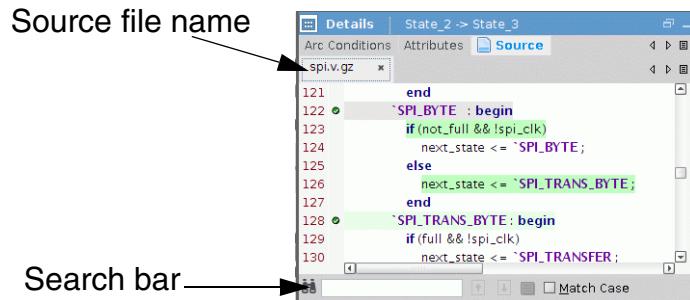
The *Col #* column of the *Attributes* pane shows the column number of the corresponding attribute. For example, a value *0* in the *Col #* column for the attribute *Name* indicates that this column appears at the first place in the *FSMs* pane. No value in the *Col #* column indicates that the corresponding attribute is not displayed in the *FSMs* pane.

For a list of all of the attributes associated with a state or transition, see [List of Attributes](#) on page 419.

Source

When you click on a state or transition in the bubble diagram or in the *States* or *Transitions* page, the corresponding source code is shown in the *Source* tab page, as shown in [Figure 2-58](#) on page 141.

Figure 2-58 Details Pane—Source Tab



In the *Source* tab page, a covered state is highlighted green and an uncovered state is highlighted red. In the above pane, all the states are covered. The same color coding is used in the coverage status locator bar. Using the coverage status locator bar, you can quickly reach the next covered or uncovered state. When you place the cursor on the coverage status locator bar, you can view the state name and the coverage count for that state. Using the search bar on the *Source* tab page, you can quickly search for a specific text in the source code. See [Performing a Search on the Source Code](#) for details on how to perform a search on the source code.

Note: The source code of a state shows the case branch where that state transitions to another state. If the state is an end state and does not transition to any other state, then the source pane will display "Nothing Shown".

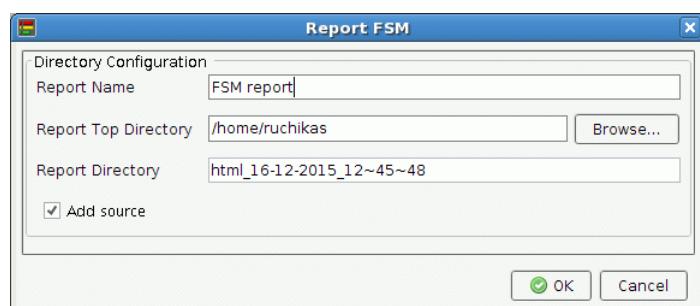
2.2.2.5 Generate HTML FSM Coverage Report

To generate an HTML FSM coverage report:

1. Click the *Report* button in the Report toolbar.

The *Report FSM* dialog box is displayed, as shown in [Figure 2-59](#) on page 141.

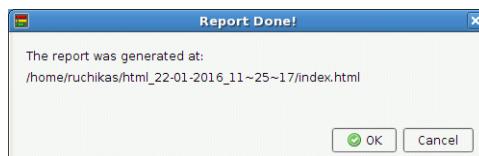
Figure 2-59 Report FSM



2. Specify the name of the report in the *Report Name* text box. By default, the name is specified as *FSM Report*. This name will appear on the HTML report.
3. In the *Report Top Directory* field, specify the location where the report must be created. By default, it shows the location from which IMC was invoked.
4. Specify the name of the report directory in the *Report Directory* text box. By default, it shows the name as `html_<timestamp>`.
5. By default, the *Add source* check box is selected. This causes reporting of *Source Code* column in the report. If you do not want to include this column in the report, clear the *Add source* check box.
6. Click *OK* to generate the report.

The *Report Done* dialog box is displayed, as shown in [Figure 2-60](#) on page 142.

Figure 2-60 Report Done



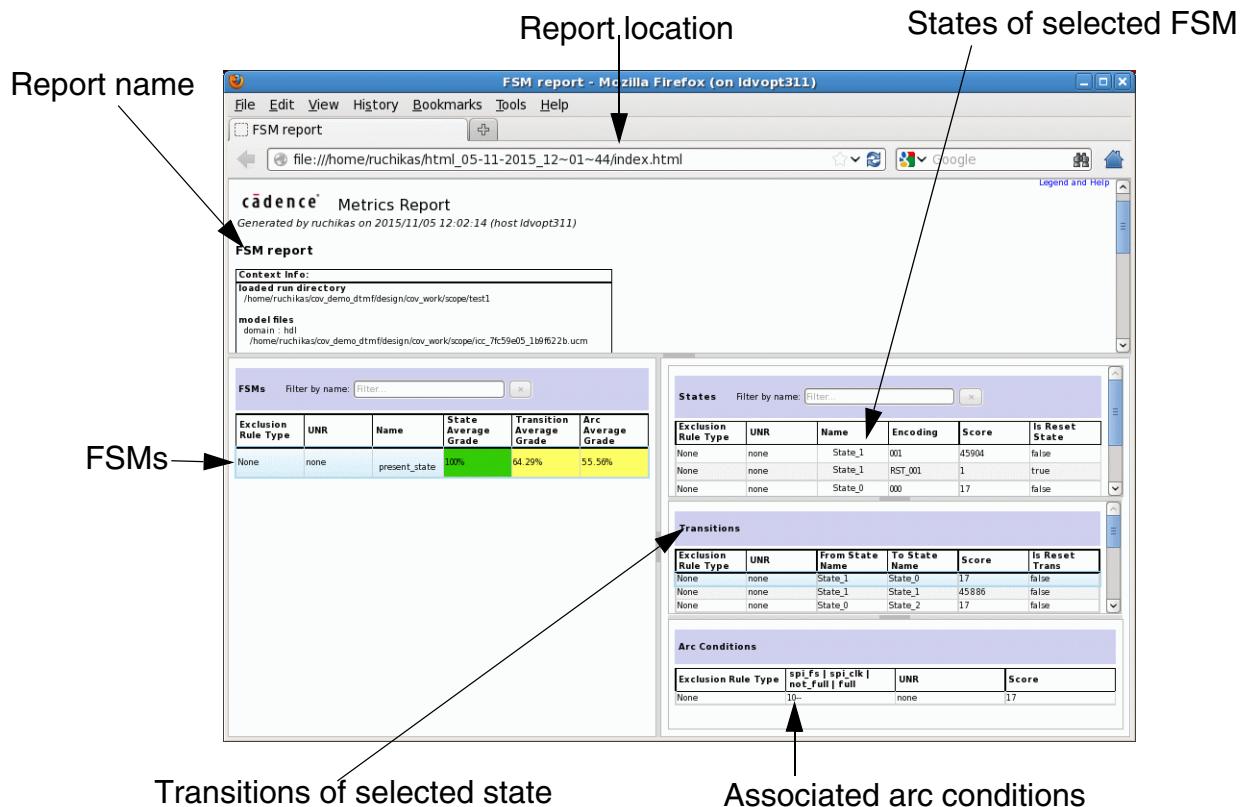
This dialog box shows the location where the report is generated.

7. Click *OK*.

You can navigate through the location shown above, and then open the `index.html` file in a Web browser.

[Figure 2-61](#) on page 143 shows the generated HTML report.

Figure 2-61 FSM Report



The report shows the information such as, who generated the report and when the report was generated. It also includes the context information, such as coverage database location, model file (.ucm) and data file (.ucd) used to generate the report, the location and contents of the coverage configuration file, and the list of refinement files applied before generating the report.

The report lists the FSM information of the selected instance or type. It shows following tables:

- **FSMs**—This table lists the FSMs within the selected instance or type.
- **States**—This table lists the states within the selected FSM.
- **Transitions**—This table lists the transitions of selected state.
- **Arc conditions**—This table lists all possible input conditions under which the transition took place.

The columns shown in the report, the sort order, and data shown in the report (filters) is based on the current state of the table at the time of generating the report. If required, you can adjust

the columns that you want to include or remove from the table, apply sort order, and also apply filters before generating the report.

Note: After you have generated the report, you can filter data based on the *Name* field using the *Filter by name* text box provided at the top of the generated HTML report.

2.3 Functional Coverage

Functional coverage is generated by inserting PSL, SystemVerilog assertions, or SystemVerilog covergroup statements into the code and simulating the design. IMC also supports analysis of functional coverage defined in **e** language and collected by Specman. The functional coverage points specify scenarios, error cases, corner cases, and protocols to be covered and also specifies analysis to be done on different values of a variable.

Functional coverage is of the following types:

- Assertion Coverage—Is an extension of assertion-based verification and identifies interesting functions directly. Assertion coverage points are specified using PSL or SVA `assert`, `assume`, and `cover` directives. The coverage to be measured is directly specified using the PSL/SVA statements or is interpreted from them.
- Covergroup Coverage—Focuses on tracking data values. It includes coverage of variable values, binning, specification of sampling, and cross products. It helps design engineers to identify untested data values or subranges. Covergroup coverage is specified using SystemVerilog constructs.

Using IMC, you can analyze both assertion as well as covergroup coverage.

2.3.1 Assertion Coverage

To analyze Assertion coverage in IMC:

1. Navigate through the design hierarchy on the *Metrics* page and identify overall coverage of different assertions in the loaded run.
2. Launch the *Assertion* page to perform a detailed analysis of different assertions.

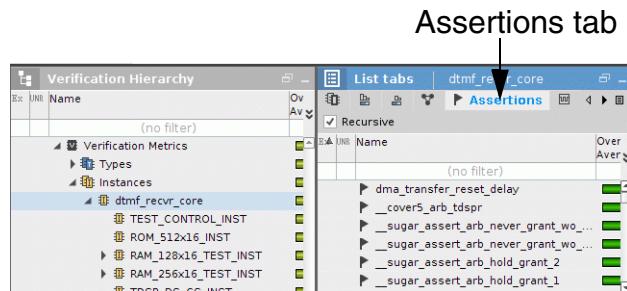
2.3.1.1 Identify Assertions for Analysis

To list all the assertions in the design and identify an assertion for analysis:

1. Click the top-level instance in the hierarchy tree.

2. Select *Assertions* tab in the right-hand pane and select the *Recursive* check box, as shown in [Figure 2-62](#) on page 145.

Figure 2-62 Metrics Page



The *Assertions* tab page displays the list of assertions along with the overall coverage of those assertions. From this list, you can identify the assertions that you want to analyze in detail and then target those assertions. Similarly, you can select a specific instance in the instance hierarchy tree and list the assertions in that instance.

2.3.1.2 Launch the Assertions Page

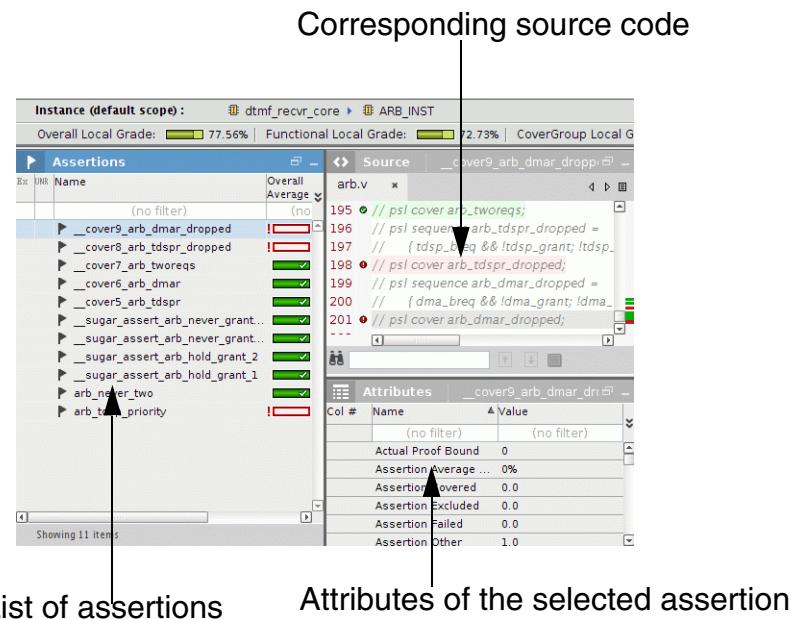
After identifying the assertions for further analysis, launch the *Assertions* page. To launch the *Assertions* page:

1. In the hierarchy tree, select the instance or type for which you want to analyze assertion coverage.
2. Click the *Assertion* icon in the *Analyze* toolbar.

Note: Alternatively, you can right-click the instance and select *Assertion Analysis*.

The *Assertions* page is launched, as shown in [Figure 2-63](#) on page 146.

Figure 2-63 Assertions Page



The *Assertions* page allows you to:

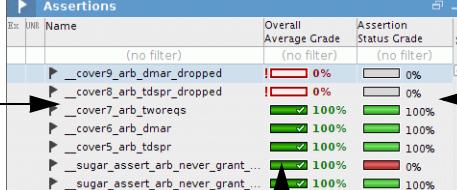
- [View the List of Assertions](#)
- [View the Underlying Source Code](#)
- [View Attributes of Selected Assertion](#)
- [View Formal, Formal_CEX, Formal_Undetermined, Formal_True_Prove, or Formal_Vacuous_Prove Items Only](#)
- [Generate HTML Assertion Coverage Report](#)

Note: The *Local* button on the *Locality* toolbar allows you to change the roll-up type. *Local* is the default, which shows assertions in the selected instance. When you click *Local*, the button is named as *Recursive*, which allows you to display assertions within all the children of the selected instance. If the button shows disabled, it indicates that you cannot change the roll-up type of the selected item.

View the List of Assertions

The *Assertions* page shown in [Figure 2-64](#) on page 147 displays the list of assertions in the selected instance or type.

Figure 2-64 Assertions Page

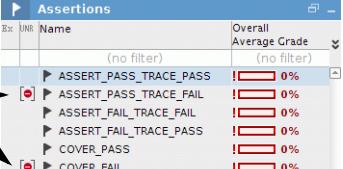


Ex: UNR	Name	Overall Average Grade	Assertion Status Grade
(no filter)	(no filter)	(no filter)	(no filter)
▶ _cover9_arb_dmar_dropped	🔴 0%	🟡 0%	
▶ _cover8_arb_tdspc_dropped	🔴 0%	🟡 0%	
▶ _cover7_arb_tworeqs	🟢 100%	🟢 100%	
▶ _covers_arb_dmar	🟢 100%	🟢 100%	
▶ _covers_arb_tdspc	🟢 100%	🟢 100%	
▶ _sugar_assert_arb_never_grant...	🟢 100%	🔴 0%	
▶ _sugar_assert_arb_never_grant...	🟢 100%	🟢 100%	

The *Assertions* page, by default, shows the assertion name, overall average grade, and status of the assertion. You can add more columns or remove columns from the table, as required. You can also filter data in the table to display only the required items, sort table data, and also search for items that meet specific search criteria. You can also filter the table data to show only excluded, covered, or uncovered items. For more details, see [View Excluded, Covered, or Uncovered Items Only](#) on page 112. You can also export the table data to a CSV file, if required. For more details, see [Export Table Data to a CSV File](#) on page 110.

If a particular assertion is marked unreachable (UNR) by Incisive Enterprise Verifier (IEV), then the *UNR* column shows an icon, as shown in [Figure 2-65](#) on page 147.

Figure 2-65 Assertions Marked Unreachable by IEV



Ex: UNR	Name	Overall Average Grade	Assertion Status Grade
(no filter)	(no filter)	(no filter)	(no filter)
▶ ASSERT_PASS_TRACE_PASS	🔴 0%		
▶ ASSERT_PASS_TRACE_FAIL	🔴 0%		
▶ ASSERT_FAIL_TRACE_FAIL	🔴 0%		
▶ ASSERT_FAIL_TRACE_PASS	🔴 0%		
▶ COVER_PASS	🔴 0%		
▶ COVER_FAIL	🔴 0%		

Note: The color of the icon and the tooltip indicates if the assertion was marked as fully unreachable or partially unreachable. A red color icon indicates fully unreachable. A fully unreachable icon shows the tooltip as **Fully Unreachable**. An orange color icon indicates partially unreachable. A partially unreachable icon shows the tooltip as **Partially Unreachable**.

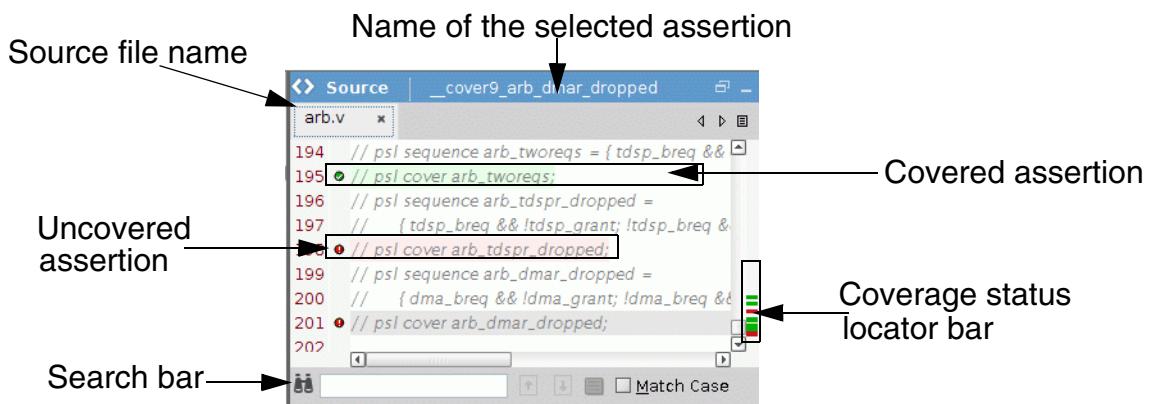
You can also filter table data to show only the items that are marked fully unreachable or partially unreachable by IEV. For more details, see [filtering UNR items](#).

For details on the UNR flow and how items are marked UNR by IEV, see the *IEV User Guide*.

View the Underlying Source Code

When you click on an assertion, the corresponding source code is shown in the *Source* pane, as shown in [Figure 2-66](#) on page 148.

Figure 2-66 Source Pane



In the *Source* pane, covered assertions are highlighted green and uncovered assertions are highlighted red. The same color coding is used in the coverage status locator bar. Using the coverage status locator bar, you can quickly reach the next covered or uncovered assertion. When you place the cursor on the coverage status locator bar, you can view the assertion name and the coverage count for that assertion.

Using the search bar on the *Source* pane, you can quickly search for a specific text in the source code. See [Performing a Search on the Source Code](#) for details on how to perform a search on the source code.

View Attributes of Selected Assertion

When you select an assertion, the corresponding attributes are shown in the *Attributes* pane. Figure 2-67 on page 149 shows the attributes of the selected assertion.

Figure 2-67 Attributes Pane

The screenshot shows a table titled "Attributes" with the assertion name "_cover9_arb_dmar_dropped" in the title bar. The table has three columns: "Col #", "Name", and "Value". The "Name" column contains attributes like "Functional Total Weights", "Functional Uncovered", "Is Active", "Is Vplan", "Name", "Overall Average Grade", and "Overall Covered". The "Value" column shows their corresponding values. Annotations with arrows point to the "Col #" column (labeled "Column number"), the "Name" column (labeled "Attribute's name"), and the "Value" column (labeled "Attribute's value").

Col #	Name	Value
	(no filter)	(no filter)
	Functional Total Weights	1.0
	Functional Uncovered	1.0
	Is Active	true
	Is Vplan	false
2	Name	_cover9_arb_dmar_dro...
3	Overall Average Grade	0%
	Overall Covered	0.0

You can right-click on an attribute and select any of the following:

- Add Columns—To add the selected column in the *Assertions* pane
- Remove Columns—To remove the selected column from the *Assertions* pane

Note: Alternatively, you can double-click on an attribute to add it to the *Assertions* pane, and then again double-click it to remove it from the *Assertions* pane.

- Unfilter Table—To release an already applied filter
- Unsort Table—To unsort the table data
- Copy Cell—To copy the data in the selected cell and paste it in any editor outside of IMC
- Copy Row—To copy the data in the selected row and paste it in any editor outside of IMC

The *Col #* column of the *Attributes* pane shows the column number of the corresponding attribute. For example, a value *1* in the *Col #* column for the attribute *Name* indicates that this column appears at the first place in the *Assertions* pane. No value in the *Col #* column indicates that the corresponding attribute is not displayed in the *Assertions* pane.

For a list of all of the attributes associated with an assertion, see [List of Attributes](#) on page 419.

View Formal, Formal_CEX, Formal_Undetermined, Formal_True_Prove, or Formal_Vacuous_Prove Items Only

The *Assertion* analysis page, provides you with the following additional views (in addition to excluded, covered and uncovered):

- Formal—To show *Ex*, *UNR*, *Name*, *Formal Average Grade*, and *Formal Status Grade* attributes in the *Assertions* pane, as shown in [Figure 2-68](#) on page 150.

Figure 2-68 Formal View

Ex	UNR	Name	Formal Average Grade	Formal Status Grade	Formal Status
		(no filter)	(no filter)	(no filter)	
		COVER_FAIL	! 0%	n/a	
		COVER_PASS	✓ 100%	n/a	
		ASSERT_PASS	✓ 100%	0%	
		ASSERT_FAIL_TRACE_PASS	✓ 100%	0%	
		ASSERT_FAIL_TRACE_FAIL	! 0%	0%	
		ASSERT_PASS_TRACE_FAIL	! 0%	0%	
		ASSERT_PASS_TRACE_PASS	✓ 100%	100%	

- **Formal_CEX**—This view shows all the attributes as shown in the *Formal* view with an additional column, *Formal Status*. This view filters the table data to show only the items for which *Formal Status* is Failed, as shown in [Figure 2-69](#) on page 150.

Figure 2-69 Formal_CEX View

Ex	UNR	Name	Formal Average Grade	Formal Status Grade	Formal Status
		(no filter)	(no filter)	(no filter)	
		ASSERT_FAIL_TRACE_PASS	✓ 100%	0%	Failed
		ASSERT_FAIL_TRACE_FAIL	✓ 100%	0%	Failed

- **Formal_Undetermined**—This view shows all the attributes shown in the *Formal* view with an additional column, *Formal Status*. This view filters the table data to show only the items for which *Formal Status* is Other and *Formal Average Grade* is > 0.0 , as shown in [Figure 2-70](#) on page 150.

Figure 2-70 Formal_Undetermined View

Ex	UNR	Name	Formal Average Grade	Formal Status Grade	Formal Status
		(no filter)	>0.0	(no filter)	~Other

- **Formal_True_Prove**—This view shows the same attributes as shown in the *Formal* view. It shows items for which *Formal Status* is Proved. This view filters the table data to show only the Proved items for which *Formal Status Grade* is > 0.0 and also the *Formal Average Grade* is > 0.0 , as shown in [Figure 2-71](#) on page 150.

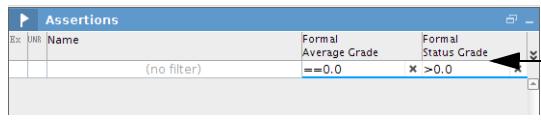
Figure 2-71 Formal_True_Prove View

Ex	UNR	Name	Formal Average Grade	Formal Status Grade	Formal Status
		(no filter)	>0.0	>0.0	

- **Formal_Vacuous_Prove**—This view shows the same attributes as shown in the *Formal* view. It shows items for which *Formal Status* is Proved. This view filters the table data

to show only the *Proved* items for which *Formal Status Grade* is >0.0 and also the *Formal Average Grade* is $= 0.0$, as shown in [Figure 2-72](#) on page 151.

Figure 2-72 Formal_Vacuous_Prove View



UN	Name	Formal Average Grade	Formal Status Grade
	(no filter)	=0.0	>0.0

Shows only the items for which Formal Status Grade > 0 and Formal Average Grade $= 0$

You can set any of the views, as required, by selecting it from the *Views* drop-down in the toolbar.

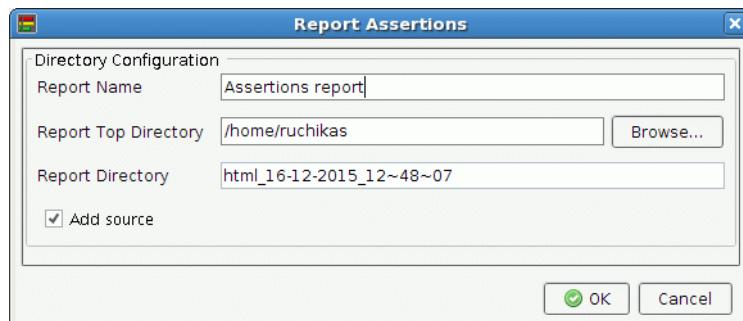
Generate HTML Assertion Coverage Report

To generate an HTML assertion coverage report:

1. Click the *Report* button in the Report toolbar.

The *Report Assertions* dialog box is displayed, as shown in [Figure 2-73](#) on page 151.

Figure 2-73 Report Assertions



2. Specify the name of the report in the *Report Name* text box. By default, the name is specified as *Assertions Report*. This name will appear on the HTML report.
3. In the *Report Top Directory* field, specify the location where the report must be created. By default, it shows the location from which IMC was invoked.
4. Specify the name of the report directory in the *Report Directory* text box. By default, it shows the name as `html_<timestamp>`.
5. By default, the *Add source* check box is selected. This causes reporting of *Source Code* column in the report. If you do not want to include this column in the report, clear the *Add source* check box.

6. Click *OK* to generate the report.

The *Report Done* dialog box is displayed, as shown in [Figure 2-74](#) on page 152.

Figure 2-74 Report Done



This dialog box shows the location where the report is generated.

7. Click *OK*.

You can navigate through the location shown above, and then open the `index.html` file in a Web browser.

[Figure 2-75](#) on page 152 shows the generated HTML report.

Figure 2-75 Assertions Report

Report.name

Report.location

List of assertions

Exclusion Rule Type	UNR	Name	Overall Average Grade	Assertion Status Grade
None	none	__cover9_arb_dmar_dropped	0%	0%
None	none	__cover8_arb_tdspr_dropped	0%	0%
None	none	__cover7_arb_tworeqs	100%	100%
None	none	__cover6_arb_dmar	100%	100%
None	none	__cover5_arb_tdspr	100%	100%
None	none	__sugar_assert_arb_never_grant_wo_req_4	100%	0%
None	none	__sugar_assert_arb_never_grant_wo_req_3	100%	100%
None	none	__sugar_assert_arb_hold_grant_2	100%	100%
None	none	__sugar_assert_arb_hold_grant_1	100%	100%
None	none	arb_never_two	100%	100%
None	none	arb_tdsp_prior	0%	0%

The report shows the information such as, who generated the report and when the report was generated. The report lists the assertions in the selected instance or type.

The columns shown in the report, the sort order, and data shown in the report (filters) is based on the current state of the table at the time of generating the report. If required, you can adjust the columns that you want to include or remove from the table, apply sort order, and also apply filters before generating the report.

Note: After you have generated the report, you can filter data based on the *Name* field using the *Filter by name* text box provided at the top of the generated HTML report.

2.3.2 Covergroup Coverage

To analyze covergroup coverage in IMC:

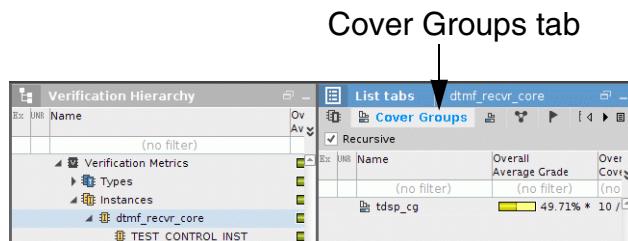
1. Navigate through the design hierarchy on the *Metrics* page and identify overall coverage of different covergroups in the loaded run.
2. Launch the *Cover Groups* page to perform a detailed analysis of different covergroups (covergroup types/covergroup instances).

2.3.2.1 Identify Covergroups for Detailed Analysis

To list all the covergroups in the design and identify a covergroup for detailed analysis:

1. Click a top-level instance in the hierarchy tree.
2. Select *Cover Groups* tab in the right-hand pane and select the *Recursive* check box, as shown in [Figure 2-76](#) on page 153.

Figure 2-76 Metrics Page



The *Cover Groups* tab page displays the list of covergroups along with the overall coverage of that covergroup. From this list, you can identify the covergroup that you want to analyze in detail and then improve its coverage.

Similarly, you can select a specific instance in the instance hierarchy tree and list the covergroups in that instance.

Note: The * appearing next to the grade in the *Overall Average Grade* column indicates that some non-default values were assigned to the weight (default is 1), goal (default is 100), or the at_least (default is 1) covergroup options. When you place the mouse over the grade value, a tooltip appears showing the values that affected the grade calculation.

2.3.2.2 Launching the Cover Group Page

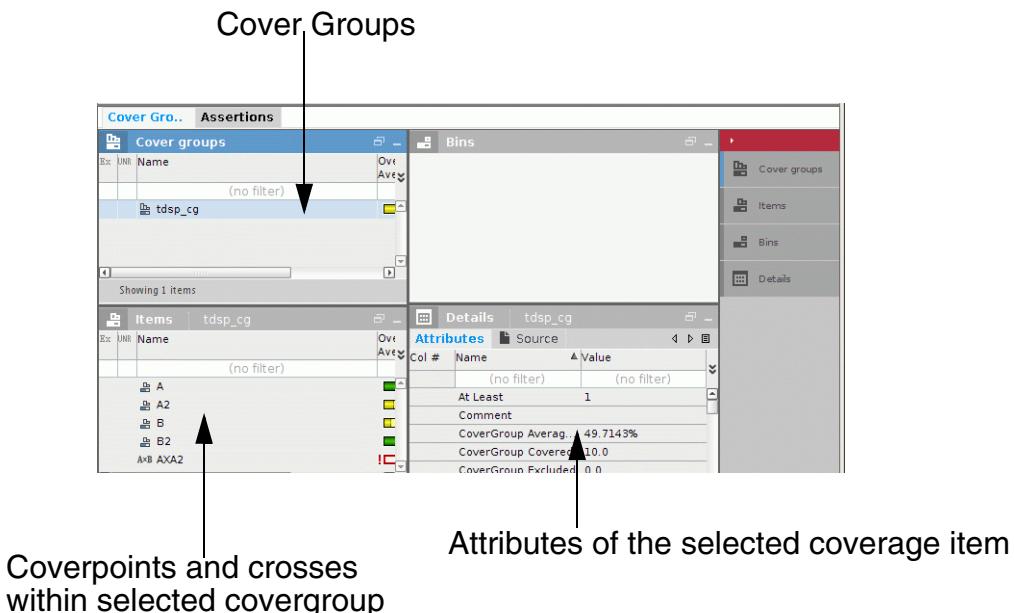
After identifying the covergroup for further analysis, launch the *Cover Groups* page. To launch the *Cover Groups* page:

1. In the hierarchy tree, select the instance or type for which you want to analyze covergroup coverage.
2. Click the *Cover Group* icon in the *Analyze* toolbar.

Note: Alternatively, you can right-click the instance and select *Cover Group Analysis*.

The *Cover Groups* page is launched, as shown in Figure 2-77 on page 154.

Figure 2-77 Cover Groups Page



The *Cover Groups* page allows you to:

- View the List of Covergroups in the selected type or instance

- [View the Cover Items](#) (coverpoints/crosses) within the selected covergroup
- [View the Cover Bins and Crosses](#) for selected cover item
- [Perform Advanced Item Analysis of Crosses](#)
- [View the Source Code and Attributes](#) of the selected covergroup or cover item
- [Generate HTML Covergroup Coverage Report](#)

Note: The *Local* button on the *Locality* toolbar allows you to change the roll-up type. *Local* is the default, which shows covergroups in the selected instance. When you click *Local*, the button is named as *Recursive*, which allows you to display covergroups within all the children of the selected instance. If the button shows disabled, it indicates that you cannot change the roll-up type of the selected item.

View the List of Covergroups

The *Cover Groups* tab page shown in [Figure 2-78](#) on page 155 displays the list of covergroups in the selected instance or type.

Figure 2-78 Cover Groups Tab Page

Ex	Unk	Name	Overall Average Grade	Overall Covered
		(no filter)	(no filter)	(no filter)
		tdsp_cg	49.71% *	10 / 56 (1...)

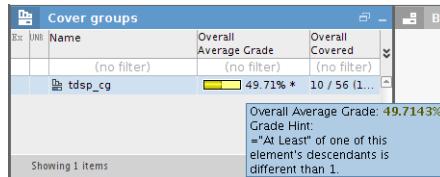
The *Cover Groups* tab page, by default, shows the name of the covergroup, overall average grade, and overall covered grade of the covergroup. You can add more columns or remove columns from the table using the [Tools & Options](#) drop-down. You can detach the pane using the *Toggle floating* option. You can hide the pane using the *Toggle auto-hide* option.

You can also [filter](#) data in the table to display only the required items, [sort](#) table data, and also [search](#) for items that meet specific search criteria. You can also filter the table data to show only excluded, covered, or uncovered items. For more details, see [View Excluded, Covered, or Uncovered Items Only](#) on page 112. You can also export the table data to a CSV file, if required. For more details, see [Export Table Data to a CSV File](#) on page 110.

Note: Sometimes, you may notice a * appearing next to the grade in the *Overall Average Grade* column. This indicates that some non-default values were assigned to the weight (default is 1), goal (default is 100), or the at_least (default is 1) covergroup options. When you place the mouse over the grade value, a tooltip appears showing the options that affected the grade calculation. The tooltip also indicates whether the options affecting the grade are related to the current entity or in one of its sibling.

Figure 2-79 on page 156 displays the tooltip that shows the options affecting the grade calculation.

Figure 2-79 Tooltip to Indicate Options Affecting Grade Calculation



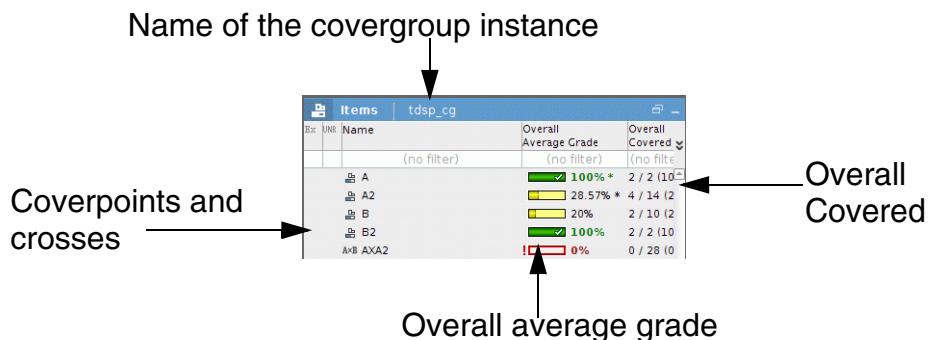
The tooltip shown indicates that non-default value was assigned to the `at_least` covergroup option.

Note: The use of non-default values assigned to the `weight`, `goal`, and the `at_least` covergroup options is not indicated in the ASCII reports of IMC.

View the Cover Items

The *Items* pane displays the coverpoints and crosses within the selected covergroup. Figure 2-80 on page 156 displays the cover items of covergroup instance `tdsp_cg`.

Figure 2-80 Cover Items



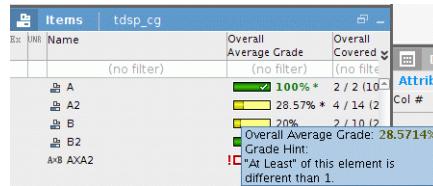
By default, the name of the coverpoints and crosses, the overall average grade, and the overall covered grade is displayed. However, you can add more columns or remove columns from the table using the Tools & Options drop-down. You can also filter data in the table to display only the required items, sort table data, and also search for items that meet specific search criteria.

Note: The * appearing next to the grade in the *Overall Average Grade* column indicates that some non-default values were assigned to the `weight` (default is 1), `goal` (default is 100),

or the at_least (default is 1) covergroup options. When you place the mouse over the grade value, a tooltip appears showing the values that affected the grade calculation.

[Figure 2-81](#) on page 157 displays the tooltip when you place the mouse over the *Overall Average Grade* column of coverpoint A.

Figure 2-81 Tooltip to Indicate Options Affecting Grade Calculation



Note: For details on the at_least option, see the *ICC User Guide and Usage and Concepts Guide for e Testbenches*.

View the Cover Bins and Crosses

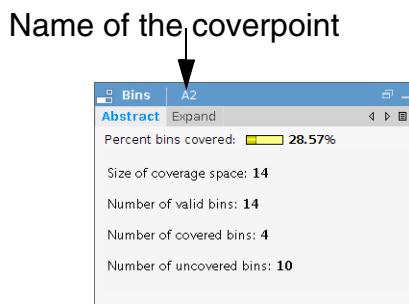
The *Bins* pane displays the bins and tuple information of the coverpoint or cross selected in the *Items* pane. It has two tab pages:

- Expand
- Abstract

By default, the *Expand* tab page is opened.

[Figure 2-82](#) on page 157 displays the *Abstract* tab page of coverpoint A2.

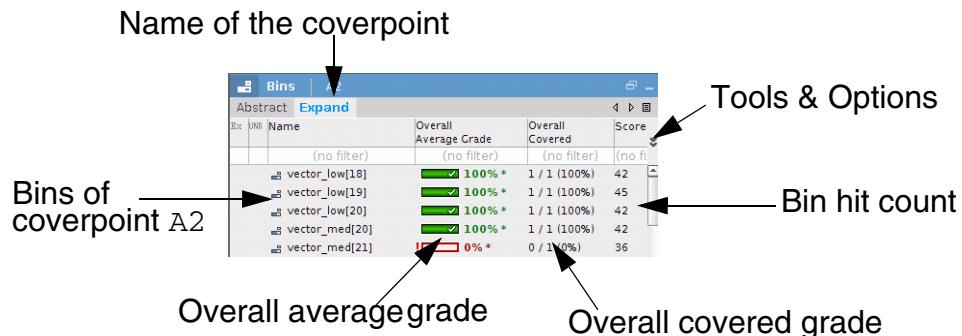
Figure 2-82 Cover Bins (Abstract)



The *Abstract* tab page shows the high-level summary of the selected cover item. The detailed information is available on the *Expand* tab page.

Figure 2-83 on page 158 displays the *Expand* tab page of coverpoint A2.

Figure 2-83 Cover Bins (Expand)



In the *Expand* tab page, by default, the name of the coverpoints and crosses, the overall average grade, overall covered grade, and the hit count of the bin are displayed. However, you can add more columns or remove columns from the table using the [Tools & Options](#) drop-down. You can also [filter](#) data in the table to display only the required items, [sort](#) table data, and also [search](#) for items that meet specific search criteria.

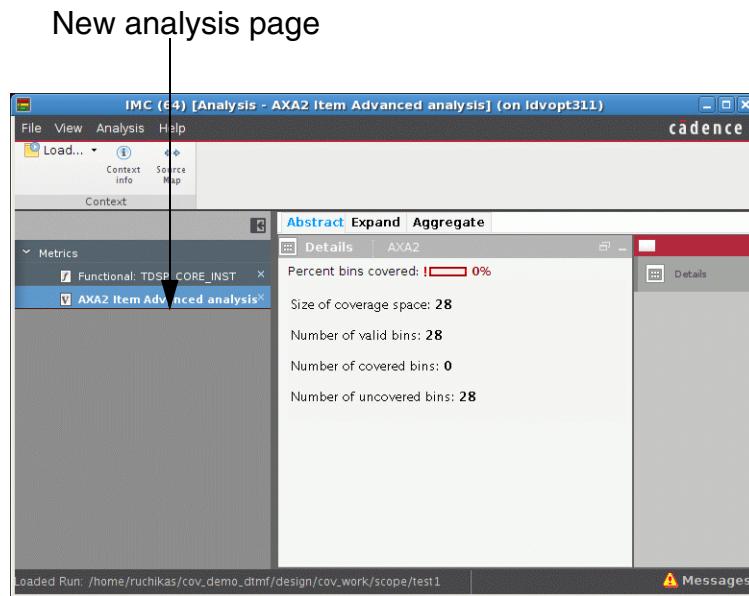
Perform Advanced Item Analysis of Crosses

To perform advanced item analysis, do any of the following:

- Select the cross, right-click and select *Item Advanced analysis*.
- Select the cross and select *Item Advanced* in the *Analyze* toolbar.

This will open a new analysis page, as shown in [Figure 2-84](#) on page 159.

Figure 2-84 Item Advanced Analysis

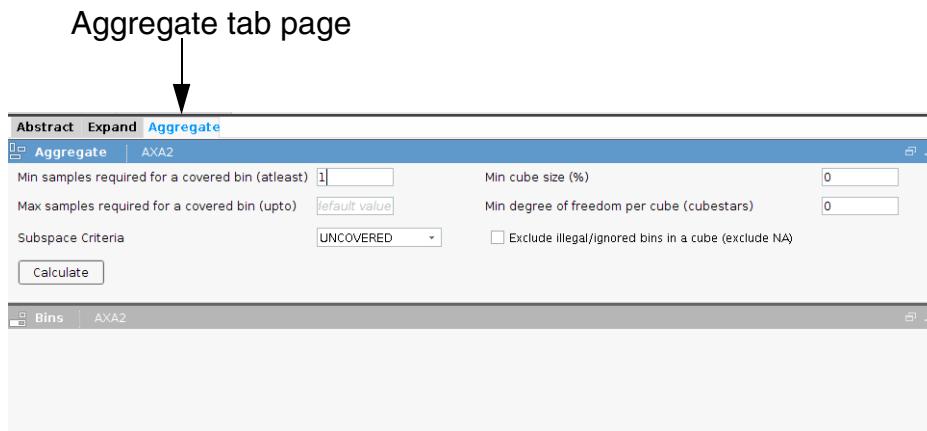


This page has following tab pages:

- The *Abstract* tab page shows the high-level summary of the selected item. For more details, see [View the Cover Bins and Crosses](#).
- The *Expand* tab page shows the list of bins and their detailed information. For more details, see [View the Cover Bins and Crosses](#).
- The *Aggregate* tab page enables generating aggregated results based on certain parameters.

[Figure 2-85 on page 160](#) shows the *Aggregate* tab page.

Figure 2-85 Item Advanced Analysis (Aggregate)



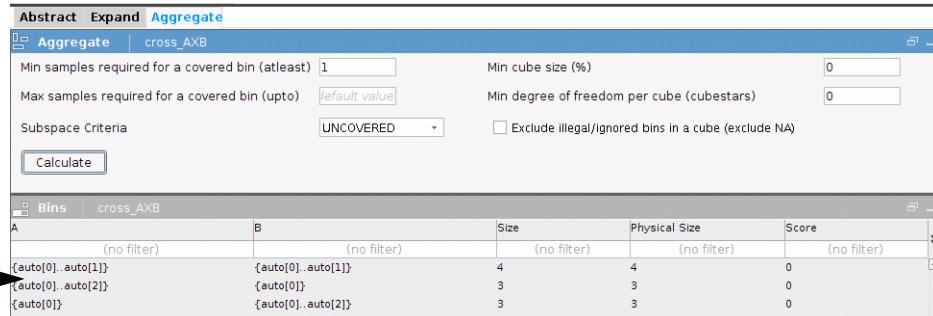
In the *Aggregate* tab page, you can set the following options:

- *Min samples required for a covered bin (atleast)*—Set the minimum number of samples required for a bin to be considered covered.
- *Max samples required for a covered bin (upto)*—Set the maximum number of samples required for a bin to be considered covered.
- *Subspace Criteria*—Specify the criteria on which the subspace creation must be based. It can be either COVERED or UNCOVERED.
- *Min cube size (%)*—Set the minimum size of reported cubes. It is measured in % of physical space size.
- *Min degree of freedom per cube (cubestars)*—Set the minimum number of stars in the reported cubes.
- *Exclude illegal/ignored bins in a cube (exclude NA)*—If selected, the NA bins are not included in reported holes/covered cubes.

After specifying the required options, click the *Calculate* button.

Figure 2-86 on page 161 shows the *Aggregate* tab page with aggregated results.

Figure 2-86 Item Advanced Analysis (Aggregated Results)



The screenshot shows the 'Aggregate' configuration window. The 'Bins' tab is selected, displaying a table of aggregated results for 'cross_AXB'. The table has columns: A, B, Size, Physical Size, and Score. The data is as follows:

A	B	Size	Physical Size	Score
(no filter)	(no filter)	(no filter)	(no filter)	(no filter)
{auto[0]..auto[1]}	{auto[0]..auto[1]}	4	4	0
{auto[0], auto[2]}	{auto[0]}	3	3	0
{auto[0]}	{auto[0]..auto[2]}	3	3	0

In the aggregated results, the similar bins are grouped together (to create cubes).

In the above results:

- Each row indicates the group of similar cross bins.
- Columns *A* and *B* indicate the coverpoints participating in the cross.

Note: These columns might sometimes show a * to indicate all values of that coverpoint. When you place the cursor over *, bins of that coverpoint (that are associated with *) are shown.

- The *Size* column shows the number of cross bins grouped together.
- The *Physical Size* column shows the physical space size taken by each group.

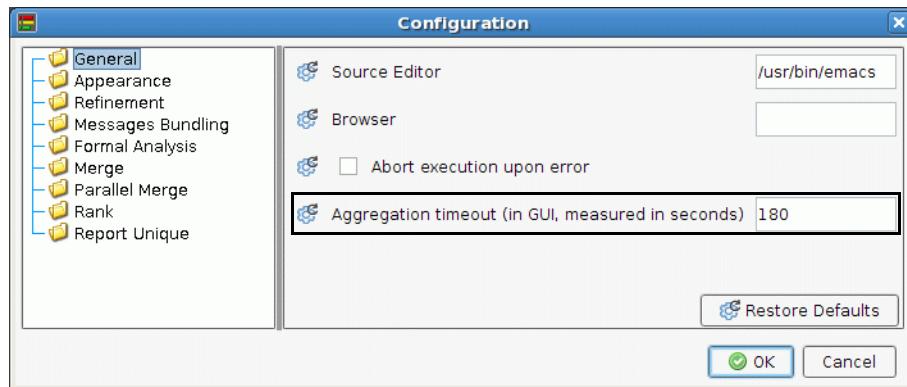
Note: A cube might include bins that are NA. This may happen if the **Exclude illegal/ignored bins in a cube (exclude NA)** option was not selected (false) at the time of calculating aggregate results. If *exclude NA* is off, then the aggregation may include NA bins inside a cube, to make the cube look nicer. Therefore, if *exclude NA* is off, you may find cubes where physical size and size are not the same.

- The *Score* column shows the number of times that group was hit.

The aggregated results enable you to quickly analyze bins. The calculation of aggregated results might increase computational overhead; however, it makes data analysis more efficient and less time-consuming.

As the calculation of aggregated results might increase the operational overhead, you can timeout the calculation after the specified time has elapsed. You can specify the timeout using the *Aggregation timeout* configuration option, as shown in [Figure 2-87](#) on page 162.

Figure 2-87 Aggregation Timeout Option



By default, the aggregation calculation is stopped after 180 seconds. However, you can change this limit, as required.

View the Source Code and Attributes

The *Details* pane of the *Cover Groups* page displays the attributes and the underlying source code of the selected cover item. The *Details* pane has the following tabs:

- Attributes
- Source

Attributes Tab

When you select a cover item, the corresponding attributes are shown in the *Attributes* tab page. [Figure 2-88 on page 162](#) shows the attributes of coverpoint bin `scalar_min`.

Figure 2-88 Attributes Page

Diagram illustrating the Attributes Page:

Col #	Name	Value
(no filter)	(no filter)	
2	Name	false
3	Overall Average Grade	100%
4	Overall Covered	1.0
	Overall Excluded	0.0
	Overall Total	1.0

Annotations:

- Name of the selected cover item: `scalar_min`
- Column number: 2
- Attribute's name: Name
- Attribute's value: false

You can right-click on an attribute and select any of the following:

- Add Columns—To add the selected column in the *Cover Groups* pane
- Remove Columns—To remove the selected column from the *Cover Groups* pane

Note: Alternatively, you can double-click an attribute to add it to the *Cover Groups* pane, and then again double-click it to remove it from the *Cover Groups* pane.

- Unfilter Table—To release an already applied filter
- Unsort Table—To unsort the table data
- Copy Cell—To copy the data in the selected cell and paste it in any editor outside of IMC
- Copy Row—To copy the data in the selected row and paste it in any editor outside of IMC

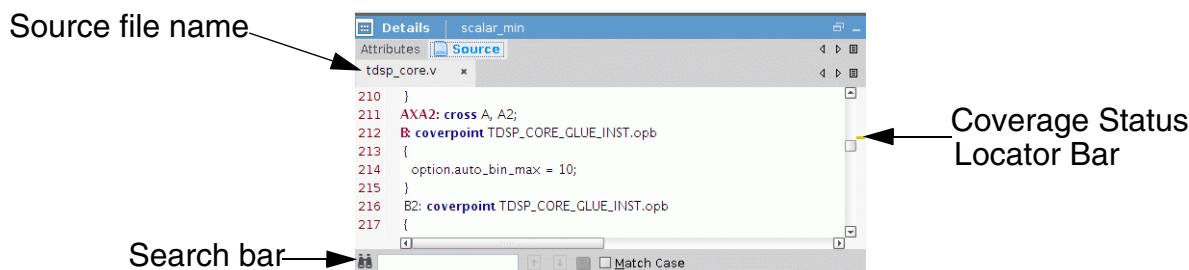
The *Col #* column of the *Attributes* pane shows the column number of the corresponding attribute. For example, a value *0* in the *Col #* column for the attribute *Atleast* indicates that this column appears at the first place in the *Cover Groups* pane. No value in the *Col #* column indicates that the corresponding attribute is not displayed in the *Cover Groups* pane.

For a list of all the attributes associated with a covergroup, coverpoint, cover bin, and cross, see [List of Attributes](#) on page 419.

Source Tab

When you click on a covergroup or any of its cover items, the corresponding source code is shown in the *Source* tab page, as shown in [Figure 2-89](#) on page 163.

Figure 2-89 Source Page



In the *Source* tab page, a covered cover item is highlighted green and an uncovered cover item is highlighted red. The same color coding is used in the coverage status locator bar. Using the coverage status locator bar, you can quickly reach the next covered or uncovered cover item. When you place the cursor on the coverage status locator bar, you can view the cover item name and the coverage count for that cover item.

Using the search bar on the *Source* tab page, you can quickly search for specific text in the source code. See [Performing a Search on the Source Code](#) for details on how to perform a search on the source code.

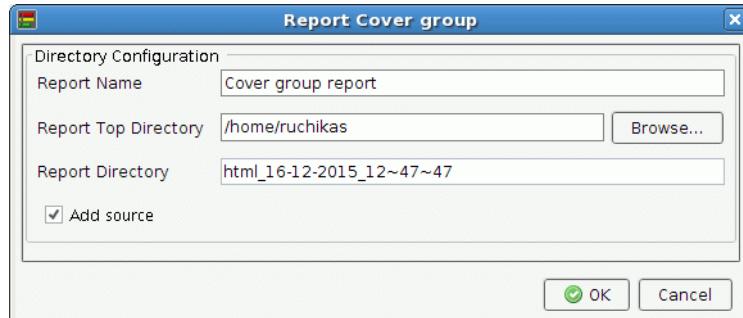
Generate HTML Covergroup Coverage Report

To generate an HTML covergroup coverage report:

1. Click the *Report* button in the Report toolbar.

The *Report Cover group* dialog box is displayed, as shown in [Figure 2-90](#) on page 164.

Figure 2-90 Report Cover group



2. Specify the name of the report in the *Report Name* text box. By default, the name is specified as *Cover group Report*. This name will appear on the HTML report.
3. In the *Report Top Directory* field, specify the location where the report must be created. By default, it shows the location from which IMC was invoked.
4. Specify the name of the report directory in the *Report Directory* text box. By default, it shows the name as `html_<timestamp>`.
5. By default, the *Add source* check box is selected. This causes reporting of *Source Code* column in the report. If you do not want to include this column in the report, clear the *Add source* check box.
6. Click *OK* to generate the report.

The *Report Done* dialog box is displayed, as shown in [Figure 2-91](#) on page 165.

Figure 2-91 Report Done



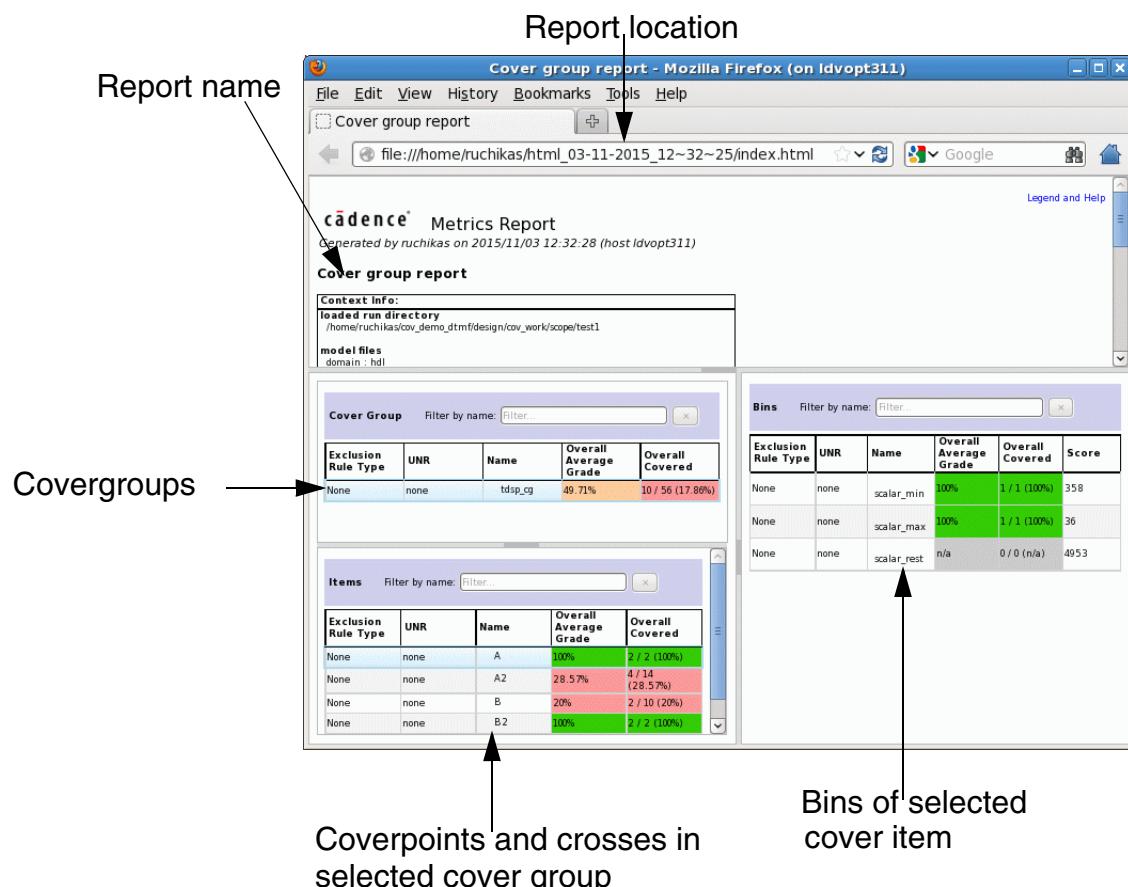
This dialog box shows the location where the report is generated.

7. Click OK.

You can navigate through the location shown above, and then open the `index.html` file in a Web browser.

[Figure 2-92 on page 165](#) shows the generated HTML report.

Figure 2-92 Cover group Report



The report shows the information such as, who generated the report and when the report was generated. It also includes the context information, such as coverage database location, model file (.ucm) and data file (.ucd) used to generate the report, the location and contents of the coverage configuration file, and the list of refinement files applied before generating the report.

The report lists the covergroup information of the selected instance or type. It shows following tables:

- Covergroup—This table lists the covergroups within the selected instance or type.
- Items—This table lists the coverpoints and crosses within the selected covergroup.
- Bins —This table lists the bins of selected coverpoint or cross.

The columns shown in the report, the sort order, and data shown in the report (filters) is based on the current state of the table at the time of generating the report. If required, you can adjust the columns that you want to include or remove from the table, apply sort order, and also apply filters before generating the report.

Note: After you have generated the report, you can filter data based on the *Name* field using the *Filter by name* text box provided at the top of the generated HTML report.

Refining Metrics Data

While analyzing coverage data, you may want to exclude certain undesired instances, types, and items from coverage analysis. The process of excluding certain items from coverage analysis is called refinement. This chapter discusses how to exclude items from coverage analysis, un-exclude excluded items, save refinements, and load the saved refinements using IMC. This chapter includes following topics:

- [Refinement in IMC GUI](#) on page 167
- [Refinement in Command-Line Interactive Mode](#) on page 188
- [Reusing Refinements File](#) on page 226

3.1 Refinement in IMC GUI

The *Refinement* toolbar contains the options that allow you to exclude instances or types, un-exclude already excluded items, save refinements, load refinements and show already excluded items. You can choose the following items from the *Refinement* toolbar:

- Exclude—To exclude the selected item and its children.
- Exclude Local Aspects only—To exclude only top-level instances.
- Exclude Smart — To exclude such that any entity connected to the excluded entity will be excluded implicitly.
- Show Affected — This option is enabled for the smart excluded entity (meaning the entity that was explicitly smart excluded). It shows the list of entities that were implicitly marked excluded because of the *Exclude Smart* action on that entity.
- Show Affecting — This option is enabled for indirect smart excluded entities (meaning any entity that was implicitly smart excluded). It shows the list of entities that caused the indirect smart exclusion.
- Edit comment—To edit exclusion comment.
- Un-Exclude—To un-exclude already excluded items.

- Clear dirty rules —To clear/delete the dirty rules. This option is available recursively from the type/instance/fsm parent entity. It is also available from the dirty excluded entity itself.
- Approve dirty rules — To approve the dirty rules (make the rule completely valid). This option is available recursively from the type/instance/fsm parent entity. It is also available from the dirty excluded entity itself.
- Clear all orphan rules—To clear/delete all the orphan rules of the selected item. This option is available recursively from the type/instance/fsm parent entity.
- Apply all orphan rules —To apply all the orphan rules of the selected item (by index of originally excluded entity) and clear them. This option is available recursively from the type/instance/fsm parent entity.
- Read Refine—To load already saved exclusion file.
- Save Refine—To save the exclusions to a file for later use.

Note: All the above options are also available in the *Analysis* menu.

3.1.1 Excluding Instances or Types

To exclude a specific instance or type from overall coverage:

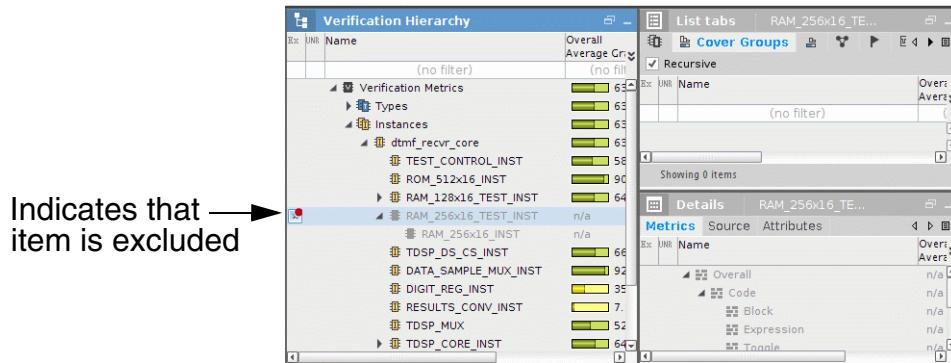
1. Select the instance or type in the verification hierarchy pane. For example, select *RAM_256x16_TEST_INST* in the verification hierarchy pane.

Note: You can also select multiple items simultaneously for exclusion using **Shift+Select** or **Ctrl+Select**. To select consecutive items, click the first item, press and hold down **SHIFT** key, and then click the last item. To select nonconsecutive items, press and hold down **CTRL**, and then click each item.

2. Select *Exclude* from the *Refinement* toolbar.

When you exclude an instance, the coverage for that instance and its child instances is removed from the overall coverage calculation, the instance is grayed out, and an icon appears along with the instance name in the instance hierarchy tree, as shown in [Figure 3-1](#) on page 169.

Figure 3-1 Refinement—Exclude



Note: Alternatively, you can exclude an instance or type by right-clicking that instance or type, and selecting *Exclude* from the pop-up menu.

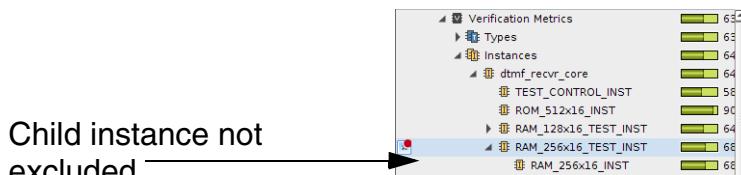
3.1.1.1 Excluding Top-Level Instance only

When excluding a top-level instance (instance that has child instances), by default, all child instances also get excluded. To *not* exclude child instances (or to exclude only the top-level instance):

- Select the instance and select *Exclude Local Aspects only* from the *Refinement* toolbar, or
- Right-click the instance and select *Exclude Local Aspects only* from the pop-up menu.

For example, to exclude only the top-level instance *RAM_128x16_TEST_INST*, right-click the instance and select *Exclude Local Aspects only* from the pop-up menu. [Figure 3-2](#) on page 169 shows that only the top-level instance is excluded.

Figure 3-2 Refinement—Exclude Local Aspects Only



3.1.2 Excluding Specific Metrics

Similar to excluding instances or types, you can exclude specific metrics for an instance or type. To exclude specific metrics for an instance or type, select that instance in the verification hierarchy pane, and then:

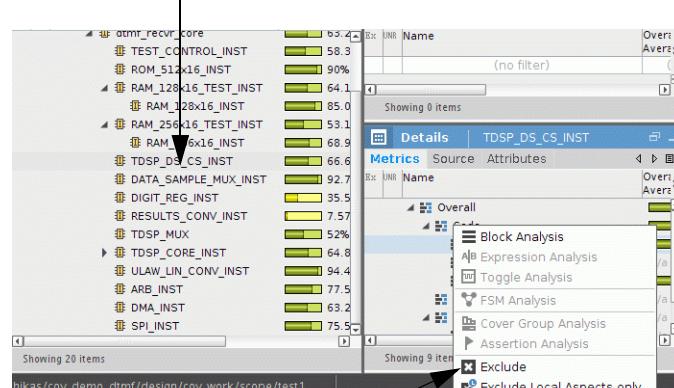
- Select the metrics type in the *Metrics* tab page, and select *Exclude* from the *Refinement* toolbar, or
- Right-click the metrics type in the *Metrics* tab page, and select *Exclude* from the pop-up menu.

For example, to exclude block metrics for the instance `TDSP_DS_CS_INST`:

1. Select `TDSP_DS_CS_INST` in the verification hierarchy pane.
2. Right-click *Block* in the *Metrics* tab page and select *Exclude* from the pop-up menu, as shown in [Figure 3-3](#) on page 170.

Figure 3-3 Refinement—Exclude Specific Metrics

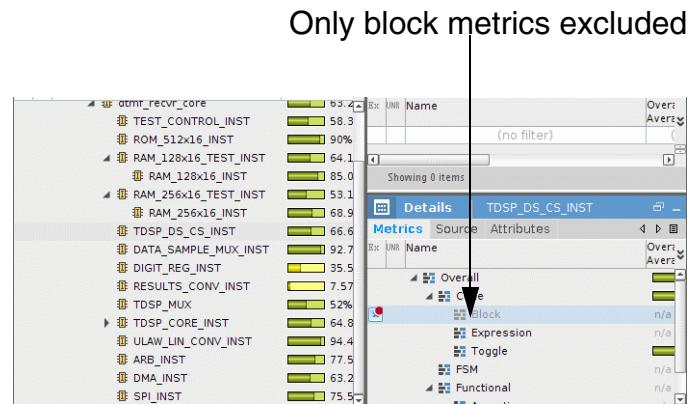
1. Select the instance.



2. Right-click metric type and select *Exclude*.

[Figure 3-4](#) on page 171 shows that only block metrics are excluded.

Figure 3-4 Refinement—Specific Metrics Excluded



Notice that only block coverage is excluded from the selected instance.

3.1.3 Excluding Specific Items

Similar to excluding instances, types, or metrics, you can exclude specific blocks, expressions, toggle signals, states, transitions, covergroups, or assertions from overall coverage. The steps for excluding specific blocks, expressions, states, transition, assertions, and covergroups are the same. This section demonstrates how to exclude specific blocks and signals.

3.1.3.1 Excluding Specific Blocks

To exclude specific blocks from overall coverage:

1. Select the block on the Block Analysis page.
2. Click *Exclude* in the *Refinement* toolbar.

Alternatively, you can:

- Right-click the block and select *Exclude* from the pop-up menu.
- Right-click the coverage icon in the *Source* tab page, and select *Exclude* from the pop-up menu.

You can also exclude multiple blocks at a time by selecting them on the *Blocks* page, and selecting *Exclude* from the *Refinement* toolbar.

Note: To select consecutive blocks, click the first block, press and hold down SHIFT key, and then click the last block. To select nonconsecutive blocks, press and hold down CTRL, and then click each block.

Similarly, you can exclude specific expressions, signals, FSMs, states, transitions, assertions, and covergroups, as required.

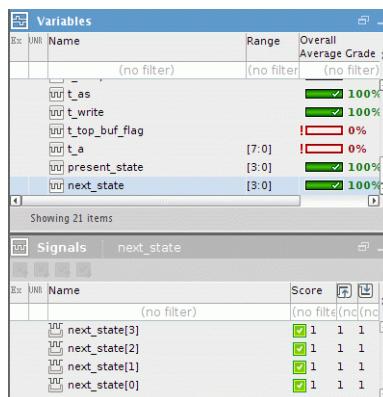
3.1.3.2 Excluding Specific Signals

In the case of toggle coverage, when you exclude a variable, all the signals associated with that variable are excluded automatically. Instead of excluding all the signals, you can consider excluding specific signals from overall coverage, as required.

To exclude specific signals from overall coverage:

1. Select the variable whose signals you want to exclude in the Variables pane on the Toggle Analysis page. For example, select *next_state* in the Variables pane, as shown in [Figure 3-5](#) on page 172.

Figure 3-5 Refinement—Exclude Specific Signals



2. The signals associated with the *next_state* are shown in the *Signals* pane. For example, select *next_state[1]* in the signals pane, right-click and select *Exclude* in the *Refinement* toolbar.

[Figure 3-6](#) on page 173 shows that signal *next_state[1]* is excluded.

Figure 3-6 Refinement—Specific Signal Excluded

Ex	UnR	Name	Score	Filter	Score
		(no filter)	(no filter)	(no filter)	(no filter)
		next_state[3]	1 1 1	<input checked="" type="checkbox"/>	1 1 1
		next_state[2]	1 1 1	<input checked="" type="checkbox"/>	1 1 1
		next_state[1]	1 1 1	<input type="checkbox"/>	1 1 1
		next_state[0]	1 1 1	<input checked="" type="checkbox"/>	1 1 1

Notice that only *next_state[1]* is excluded.

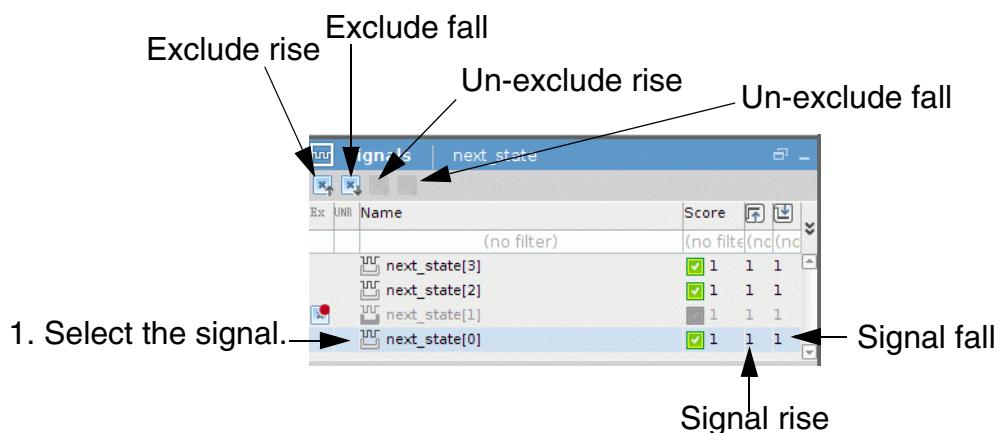
3.1.3.3 Excluding Partial Signals (Rise or Fall Transitions)

A signal bit has two states, 0 and 1. A signal is considered covered if it completes a full transition that is from 0 \rightarrow 1 (rise) and from 1 \rightarrow 0 (fall). At times, only either rise or fall transition is expected. Therefore, in such cases, it is important that instead of excluding the complete signal, we could exclude rise transition or the fall transition, as required.

To exclude partial signals (that is either rise transition or fall transition):

1. Select the signal in the *Signals* pane. For example, select *next_state[0]*, as shown in Figure 3-7 on page 173.

Figure 3-7 Refinement—Exclude Partial Signals

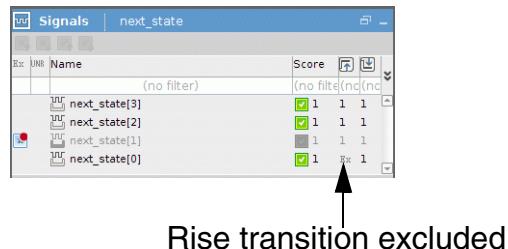


2. To exclude the rise or fall transition, select the appropriate icons from the toolbar in the *Signals* pane. For example, to exclude the rise transition, click the *Exclude rise* button in the toolbar.

Note: Alternatively, you can right-click the signal name and select *Exclude Rise* from the pop-up menu. The icons shown in the toolbar for excluding and un-excluding rise or fall transitions are also shown in the pop-up menu when you right-click the signal.

[Figure 3-8 on page 174](#) shows that for signal *next_state[0]* only rise transition is excluded.

Figure 3-8 Refinement—Only Rise Transition Excluded



Notice that only rise transition of the *next_state[0]* is excluded. Similarly, you can also exclude other transitions, as required.

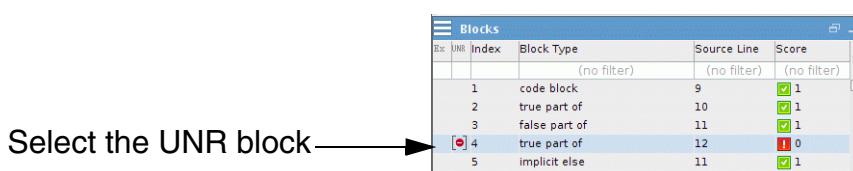
Note: If you exclude both rise as well as fall transition of a signal, the complete signal is excluded.

3.1.3.4 Excluding UNR Items

To exclude a UNR item from overall coverage:

1. Select the UNR item on the Analysis page. For example, select *Block 4* on the Block Analysis page, as shown in [Figure 3-9 on page 174](#).

Figure 3-9 Refinement—Exclude UNR Blocks

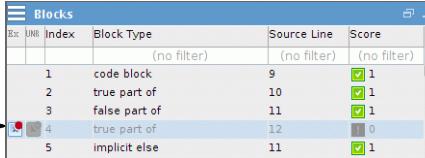


2. Right-click the block and select *Exclude* from the pop-up menu.

Note: Alternatively, you can click *Exclude* in the *Refinement* toolbar.

[Figure 3-10 on page 175](#) shows Block 4 marked excluded.

Figure 3-10 Refinement—Exclude UNR Blocks



UNR block marked excluded →

Ex	UNR	Index	Block Type	Source Line	Score
			(no filter)	(no filter)	(no filter)
		1	code block	9	<input checked="" type="checkbox"/> 1
		2	true part of	10	<input checked="" type="checkbox"/> 1
		3	false part of	11	<input checked="" type="checkbox"/> 1
		4	true part of	12	<input type="checkbox"/> 0
		5	implicit else	11	<input checked="" type="checkbox"/> 1

For details on excluding UNR items in CLI mode, see [Excluding Unreachable Items](#) on page 221.

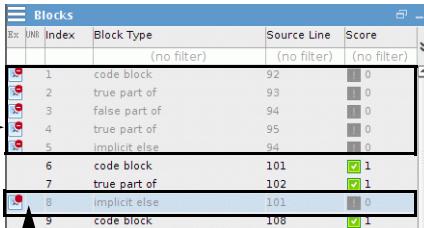
3.1.3.5 Difference Between Manual Marking and Automatic Marking

Manual marking means to mark certain items manually at the time of analysis. For example, you select a specific block in IMC, and remove it from analysis by marking it as *Exclude*.

There are certain items that are automatically marked *Exclude* at the time of simulation run.

[Figure 3-11](#) on page 175 shows some items marked manually and some items marked automatically at the time of simulation run.

Figure 3-11 Refinement—Manual Marking and Automatic Marking



Items marked automatically (at the time of simulation run) →

Item marked manually (at the time of analysis) →

Ex	UNR	Index	Block Type	Source Line	Score
		1	code block	92	<input type="checkbox"/> 0
		2	true part of	93	<input type="checkbox"/> 0
		3	false part of	94	<input type="checkbox"/> 0
		4	true part of	95	<input type="checkbox"/> 0
		5	implicit else	94	<input type="checkbox"/> 0
		6	code block	101	<input checked="" type="checkbox"/> 1
		7	true part of	102	<input checked="" type="checkbox"/> 1
		8	implicit else	101	<input type="checkbox"/> 0
		9	code block	108	<input checked="" type="checkbox"/> 1

Notice the slight difference between the icons used for automatically marked items and the items marked manually.

Using IMC, you can also apply filters to:

- View only the items marked automatically at the time of simulation run
- View items marked manually during analysis

Note: The row below the column header is the filter row in which you can specify the filtering criteria. For more details on filtering, see [Filtering Values in Ex Column](#) on page 68.

3.1.4 Excluding Connected Entities (Smart Refinement)

In a coverage model, there are entities that might have connected entities. For example, a cover item participating in a cross. In such situations, if the cover item is excluded, then the cross (which is the connected entity) should also be excluded implicitly. Such exclusion is called smart exclusion.

The *Exclude Smart* refinement option allows you to apply smart exclusions, that is, any entity connected to the excluded entity will be excluded implicitly.

Currently, smart exclusion is supported in following scenarios:

- Smart exclusion of a toggle variable causes exclusion of all of the associated signals.
- Smart exclusion of a multi-driver signal causes an implicit exclusion of signal bits connected to it.
- Smart exclusion of a signal bit causes exclusion of signal bits connected to it.
- Smart exclusion of rise or fall transition of a signal bit causes exclusion of associated signals.
- Smart exclusion of a signal enum causes an implicit exclusion of connected enum signals.

Note: Propagation of exclusion from enum nets and enum values is done only for an exact match of both driver information (same number of bits and driven by same drivers) and enum names. In addition, exclusion of enum nets propagates only to enum nets and exclusion of non-enum nets propagates to non-enum nets. If an enum net is connected to a non-enum net and both might have the same width and might be driven by the same drivers, exclusion of one will not be propagated to other.

- Smart exclusion of a state excludes any transitions to/from that state.
- Smart exclusion of all transitions to/from a state excludes that state implicitly.
- Smart exclusion of a bin excludes all corresponding bins from relevant crosses.
- Smart exclusion of a cover item excludes any cross that contains it as one of its dimensions.

Smart exclusion is currently not supported in following cases:

- It is not supported for cover item in a covergroup that is contained in a cross of another covergroup.
- Smart exclusion is currently not supported for blocks, expressions, and min-term for expressions.

Points to Remember

- Results might change based on the use of `-expand` switch at elaboration.
 - Without the `-expand` switch, only signals (not signal-bits) can be smart-excluded or smart indirectly excluded. In such cases, all of the signal-bits are `P-EXCL` from the signal.
 - With the `-expand` switch, both signals as well as signal-bits can be smart-excluded. In such cases, the smart exclusion propagates through the signal-bits and not through the signals.
- Results might change based on the use of `-LPS_ASSIGN_FT_BUF` switch (to disable treating continuous assignment as pass through net) at elaboration.

Note: Smart exclusion is supported only for analysis time exclusions.

This section covers following topics:

- [Applying Smart Exclusion](#)
- [Viewing Impacted Entities](#)
- [Viewing Impacting Entities](#)

3.1.4.1 Applying Smart Exclusion

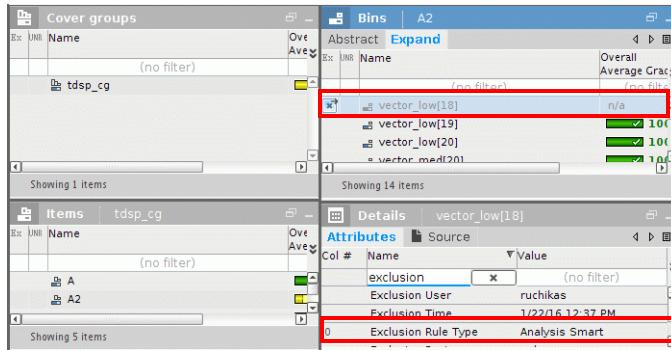
To apply smart exclusion, perform the following steps:

1. Select the item on which you want to apply smart exclusion. For example, select `vector_low[18]` bin of cover item `A2`.
2. Right-click and select *Exclude Smart*.

Note: Alternatively, you can click *Exclude Smart* option on the Refinement toolbar or select *Exclude Smart* from the Analysis menu.

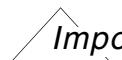
The selected item is marked excluded, as shown in [Figure 3-12](#) on page 178.

Figure 3-12 Smart Exclusion



When you apply smart exclusion, the value of *Exclusion Rule Type* attribute is changed to *Analysis Smart*.

The value of *Exclusion Rule Type* attribute for the entities that are connected to the smart excluded entity is *Analysis Smart Indirect* (because they got excluded indirectly due to a *Exclude Smart* action).

 **Important**

The *Unexclude* action on an entity that is smart excluded will also unexclude all its connected entities. However, if you perform an unexclude action on an entity that got excluded indirectly due to a smart exclude action will not cause the master entity to unexclude. For example, entity *a1* is the master entity and smart exclude action on it excludes both *a1* and *a2* (because *a2* is a connected entity). If you perform an unexclude on *a1*, it will unexclude both *a1* and *a2*. But, if you perform an unexclude action on *a2*, it will only exclude *a2* and not the master entity that is *a1*. In addition, remember that this unexclude on indirect will not work for signal rise/fall.

3.1.4.2 Viewing Impacted Entities

After you have applied smart exclusion on an item, you can view the entities that were impacted due to exclusion (that is view the list of entities that were connected to the excluded entity).

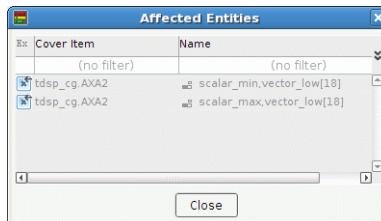
For this:

1. Select the excluded entity.
2. Right-click and select *Show Affected*.

Note: Alternatively, you can click *Show Affected* option on the Refinement toolbar or select *Show Affected* from the Analysis menu.

A dialog box is opened which shows the affected entities, as shown in [Figure 3-13](#) on page 179.

Figure 3-13 Affected Entities



The affected entities are listed in the dialog box.

3.1.4.3 Viewing Impacting Entities

If an entity is indirect smart excluded (that is -- it is excluded implicitly due to a *Exclude Smart* action), you can view the list of entities that caused the indirect smart exclusion.

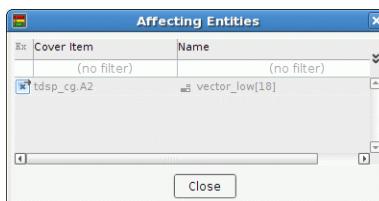
For this:

1. Select the Indirect smart excluded entity.
2. Right-click and select *Show Affecting*.

Note: Alternatively, you can click *Show Affecting* option on the Refinement toolbar or select *Show Affecting* from the Analysis menu.

A dialog box is opened which shows the affecting entities, as shown in [Figure 3-13](#) on page 179.

Figure 3-14 Affecting Entities



The affecting entities are listed in the dialog box.

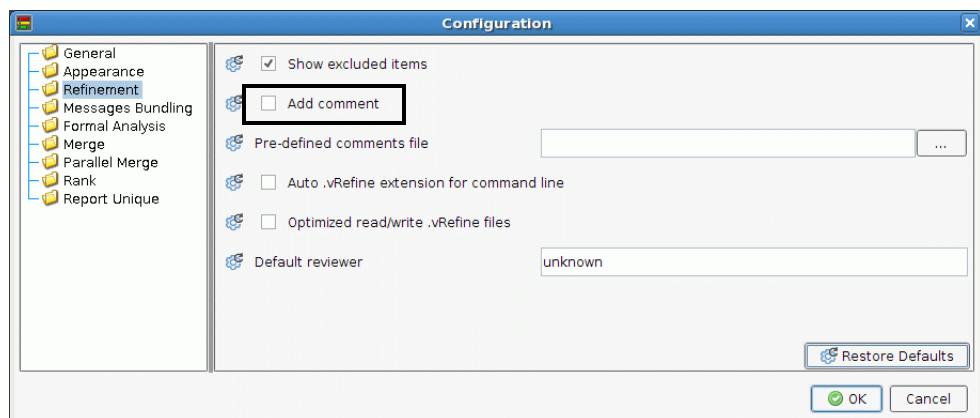
3.1.5 Adding Comments at the Time of Exclusion

When excluding an instance, type, or any coverage item, you can add comments to describe the refinement.

For this, before marking a particular item as excluded, you must enable adding comments.

You can do this by setting the *Add comment* option in the *Configuration* dialog box, as shown in [Figure 3-15](#) on page 180.

Figure 3-15 Configuration—Refinement Options



By default, the *Add comment* check box is not selected. This indicates that you will not be prompted to add comments at the time of exclusion. After you select the *Add comment* check box, you will be prompted to add comments at the time of exclusion. A dialog box will be displayed which will prompt you to add comments.

Note: Once you enable the *Add comment* check box, it applies to all the exclusions done in that invocation of IMC.

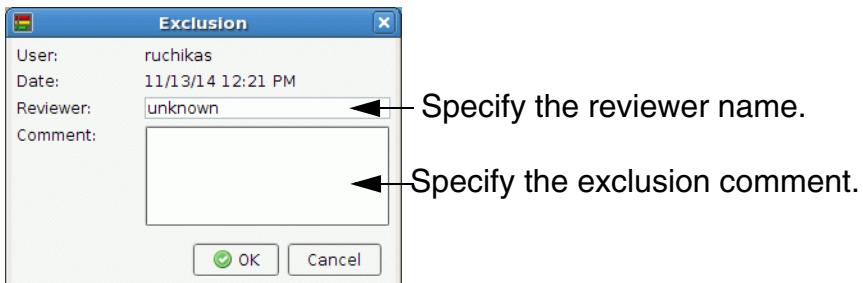
Consider an example. To mark the instance TDSP_CORE_INST as exclude and to add user comments at the time of exclusion, perform the following steps:

1. Select *Configuration* from the *View* menu.
2. Click the *Refinement* folder.
3. Select the *Add comment* check box and click *OK*.

Note: Once you enable the *Add comment* check box, it applies to all the exclusions done in that invocation of IMC. As a result, steps 1, 2, and 3 need not be repeated for each exclusion.

4. Select the instance or type in the verification hierarchy pane. For example, select *TDSP_CORE_INST* in the verification hierarchy pane.
5. Select *Exclude* from the *Refinement* toolbar. When you click the *Exclude* button, the *Exclusion* dialog box is displayed, as shown in [Figure 3-16](#) on page 181 for adding user comments.

Figure 3-16 Refinement—Exclusion with Comments



In the *Exclusion* dialog box, you can specify the name of the reviewer and also the comments to describe the reason for excluding the specified entity. The *Exclusion* dialog box also displays two non-editable fields, *User* and *Date*. These values are picked automatically from the system on which the tool is running.

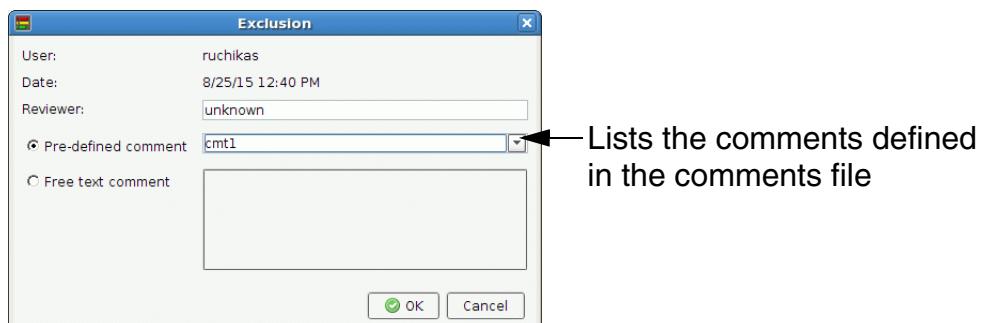
6. By default, the reviewer is specified as *unknown*. To specify the reviewer name as *Andrew*, delete the text *unknown*, and specify *Andrew* in the *Reviewer* field. You can also set a default reviewer once in the *Configuration* dialog box and it will appear automatically in the *Exclusion* dialog box. By doing so, you will not have to enter it every time you make exclusion. For more details, see [Configuring Refinement Options](#) on page 81.

Note: The reviewer name cannot be more than 24 characters.

7. In the *Comments* field, specify the comment. For example, specify the comment as, *The instance seems irrelevant for analysis*. Comment text cannot be more than 500 characters.

Note: In case you have specified a comments file in the *Pre-defined comments file* of the *Configuration* dialog box, you will see additional fields, as shown in [Figure 3-17](#) on page 182.

Figure 3-17 Refinement—Exclusion from Comments File

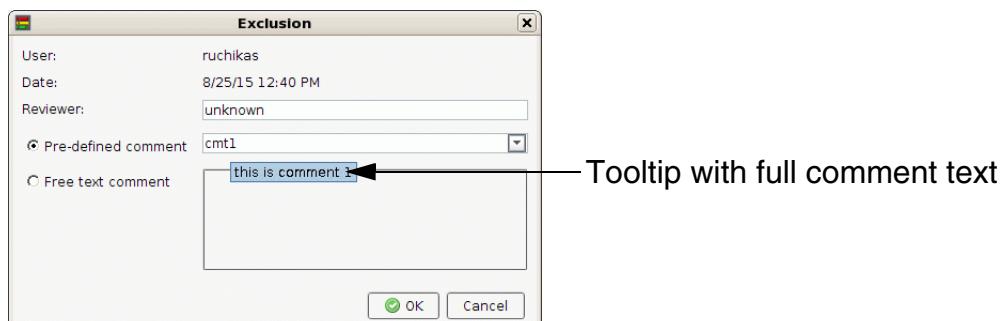


The *Pre-defined comment* drop-down lists all the comments that are defined in the comments file (that you specified in the Configuration dialog box). You can place the cursor on the comment name that appears in the drop-down list. It will show you the associated comment text as a tooltip. In this case, the underlying comments file has the following text:

```
NAME,COMMENT
cmt1,this is comment 1
cmt2,this is comment 2
```

When you place the cursor on *cmt1* in the *Exclusion* dialog box, the associated comment is displayed, as shown in as shown in [Figure 3-18 on page 182](#).

Figure 3-18 Refinement—Exclusion from Comments File



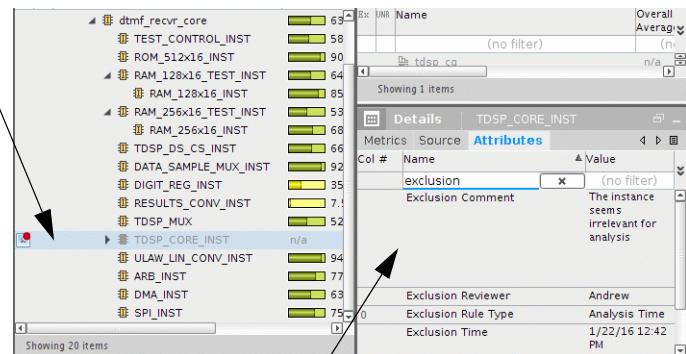
In case you do not want to add comment from the pre-defined list of comments, click the *Free text comment* radio button and specify the comment in the text field.

8. Click *OK*.

After performing the above steps, the instance *TDSP_CORE_INST* is excluded. In addition, the values specified in the *Exclusion* dialog box are displayed in the *Attributes* tab page, as shown in [Figure 3-19 on page 183](#).

Figure 3-19 Adding User Comments

Instance marked excluded



Exclusion comment and Exclusion reviewer

You can add the required attributes to the table shown in the *Verification Hierarchy* pane and the *List tabs* pane, as required. For details, on adding columns in tables, see [Adding Columns](#) on page 64. Once you add the attributes to required tables, you can apply filters, sort data, and also save it as a view. For more details, see [Defining and Organizing Views](#) on page 72.

Note: You can also add comments for multiple coverage items simultaneously. When adding comments for multiple items, the comments added will apply to all of the selected entities.

Note: The comments added at the time of exclusion are preserved and are available for later use if the exclusions are saved to a refinement file. For more details on saving a refinement file, see [Saving Refinements](#) on page 186. In addition, the exclusion comments and reviewer details added and saved in IMC GUI are available in the batch mode of IMC and vice versa.

Note: In case you add comments at the time of smart exclusion, then the comment you add to the entity is also applied to all of its connected entities. If the connected entity is getting excluded from more than one connections, then the comment information of the first origin is shown. For details on smart exclusion, see [Excluding Connected Entities \(Smart Refinement\)](#) on page 176.

3.1.6 Editing User Comments

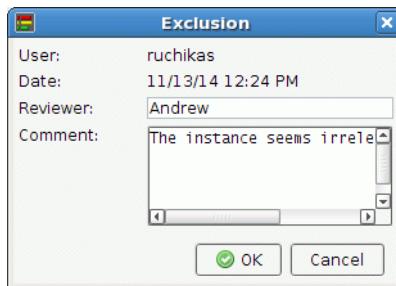
After adding user comments, you can edit the comments, as required.

To edit user comments, select the excluded item whose comments you want to edit, and then click the *Edit Comment* button on the *Refinement* toolbar.

Note: Alternatively, you can select the excluded item whose comments you want to edit, right-click and select *Edit Comment for exclusion* from the pop-up menu.

This opens the *Exclusion* dialog box, as shown in [Figure 3-20](#) on page 184.

Figure 3-20 Editing User Comments

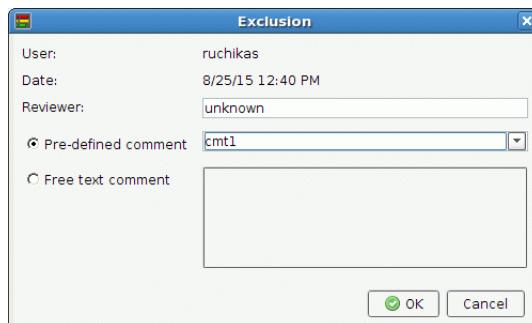


You can edit the reviewer and comment, as required, and click on *OK*.

You can also delete the reviewer and comment by removing the values in the respective fields.

Note: In case you have specified a comments file in the *Pre-defined comments file* of the *Configuration* dialog box, you will see additional fields in the *Exclusion* dialog box at the time of editing comments, as shown in [Figure 3-21](#) on page 184.

Figure 3-21 Refinement—Editing Comments



Note: If an old refinement file (from previous releases) is loaded, then at the time of editing user comments, the *User* and *Reviewer* fields show the value as *unknown*, *Date* field is set to default 01/01/1970, and the *Comment* field does not show any value.

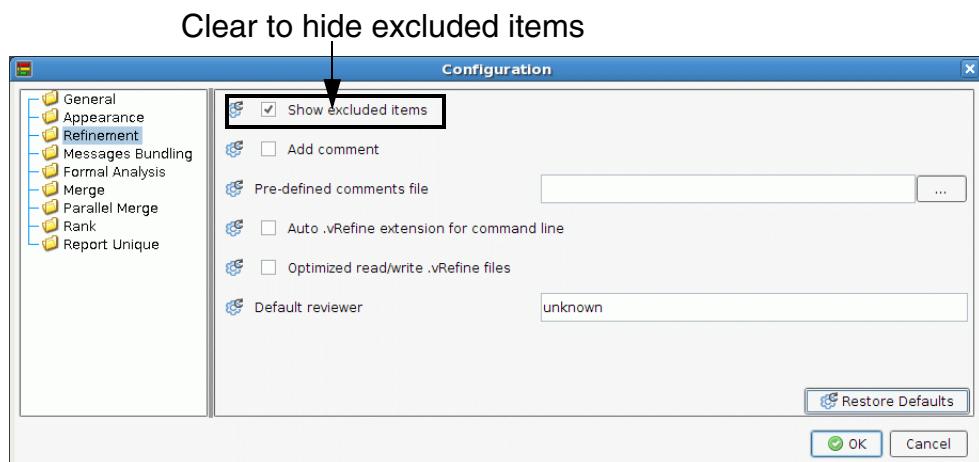
3.1.7 Hide Excluded Items

When you exclude an instance, type, or specific metrics for an instance, then by default, that instance or type still appears in the hierarchy (though the coverage for that item is excluded)

from overall coverage). However, you can hide the excluded instance or types and not show those items in the hierarchy.

For this, you need to clear the *Show excluded items* check box in the *Configuration* dialog box, as shown in [Figure 3-22](#) on page 185.

Figure 3-22 Configuration—Hide Excluded Items



Note: You can invoke the *Configuration* dialog box by selecting *Configuration* from the *View* menu. For refinement related options, select *Refinement* in the left pane.

After you clear the *Show excluded items* check box, the excluded items will not be shown in the verification hierarchy.

3.1.8 Un-Excluding Excluded Items

To un-exclude an already excluded item:

1. Select the excluded instance or type in the verification hierarchy pane. For example, select *RAM_128x16_TEST_INST* in the verification hierarchy pane.
2. Click *Un-Exclude* in the *Refinement* toolbar.

Note: Alternatively, you can right-click the instance or type and select *Un-Exclude* from the pop-up menu.

After selecting *Un-Exclude*, the coverage for that instance is included in the overall coverage.

Similarly, you can un-exclude specific coverage metrics, as required.

Note: You can also un-exclude multiple items, such as blocks, expressions, FSMs, states, transitions, assertions, and covergroups at a time by selecting them on specific analysis pages, and selecting *Un-Exclude* from the *Refinement* toolbar.

3.1.9 Saving Refinements

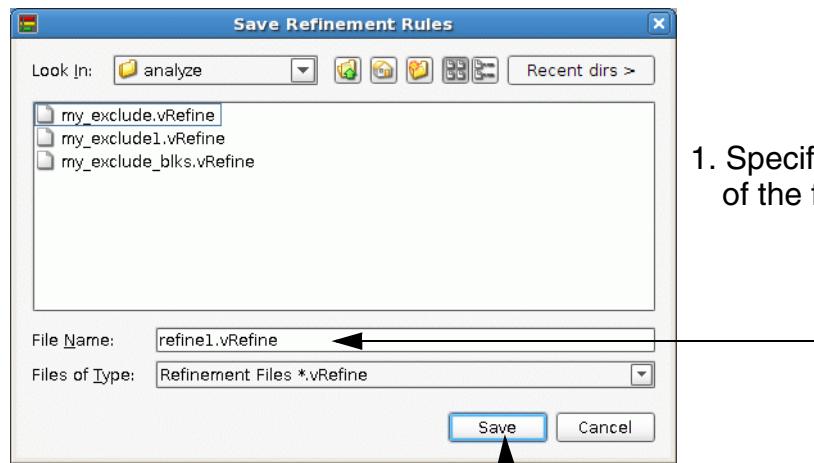
After excluding coverage items or instances during analysis, you can save the refinements to a file and later, in another IMC run, load the saved refinements. This helps in saving the effort of making exclusions again in the next IMC run.

To save the refinements made in one IMC run for reuse in another run:

1. Click *Save Refinement Rules to File* in the *Refinement* toolbar.

The Save Refinement Rules dialog box appears, as shown in [Figure 3-23](#) on page 186.

Figure 3-23 Save Refinement Rules



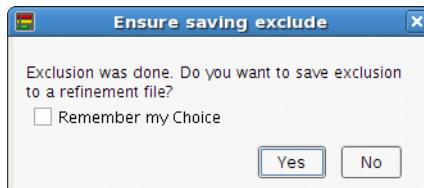
2. By default, file name appears as `refine1.vRefine`. You can specify a different name for the refinement file by specifying the name in the *File Name* text box and click *Save*. For example, specify the name as `my_exclude_blk.vRefine`.

Note: The extension of a refinement file is always `.vRefine`.

In this case, when you save the file, the file will be named as `my_exclude_blk.vRefine`.

Note: If you have applied some refinements and did not save the refinement, then at the time of exiting IMC, a dialog box will appear, as shown in [Figure 3-24](#) on page 187 to indicate that refinements were made and if you want to save the refinement before exiting the application.

Figure 3-24 Refinement—Ensuring Saving of Refinements



This will ensure that you do not miss saving the refinement before you exit IMC. You can:

- Click **Yes** to save the refinement before exiting.
- Click **No** to not save the refinement before exiting.

Note: You can also select the *Remember my Choice* check box to ensure that your choice (Yes or No) is saved and preserved for reuse in subsequent similar situations.

Note: The refinements saved in GUI mode can be loaded and applied in the batch mode of IMC and vice versa. In addition, the refinement file saved with 12.2 version of IMC will not be supported in any prior version of IMC.

3.1.10 Loading Saved Refinements

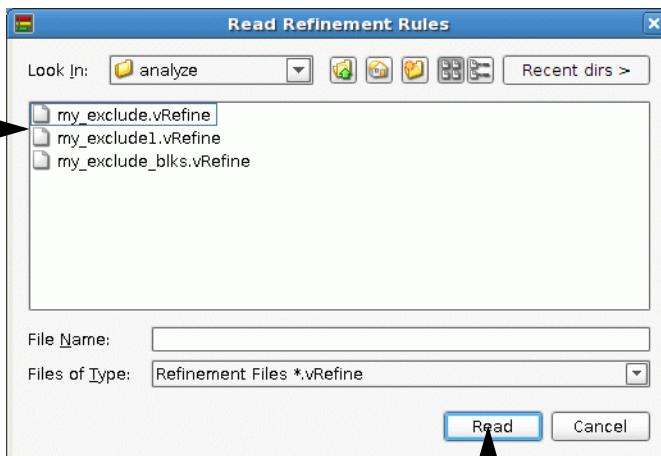
To load the already saved refinements for reuse in another IMC run:

1. Click *Refinement* and select *Read Refinement Rules from File*.

The Read Refinement Rules dialog box appears, as shown in [Figure 3-25](#) on page 187.

Figure 3-25 Read Refinement Rules

1. Select the file.



2. Click **Read**.

2. Select the refinements file you want to load, and click *Read*.

When you load the refinements file, the saved exclusions are applied to the loaded run.

3.2 Refinement in Command-Line Interactive Mode

The `exclude` command of IMC enables you to exclude metrics items in command-line interactive mode.

3.2.1 Command Syntax

The complete BNF of the `exclude` command is:

```
exclude [-type <type> | -inst <instance>]
[-metrics <metrics_type>] | <coverage_object_specification>
[-smart]
[-unr]
[-comment <comment>]
[-reviewer <reviewer_name>]
```

where

```
<metrics_type> ::= [all] [block] [expression] [toggle] [fsm] [assertion]
[covergroup] [state] [transition] [arc] [code] [functional]
<coverage_object_specification> ::= -assertion <assertion_name> |
-covergroup <covergroup_name> |
-coveritem <covergroup_name>. <coverpoint_name> |
-coverbin <covergroup_name>. <coverpoint_name>. <coverbin_name> |
-fsm <fsm_name> |
-state <fsm_name>. <state_name> |
-transition <fsm_name>. <from_state>/ | . <to_state> |
-arc <fsm_name>. <from_state>. <to_state>. <arc_index>
-toggle <signal_full_name> [[<bit_specification>] [.rise | .fall]] |
-block <index_specification> |
-expression <index_specification>

<index_specification> ::= <index> | <range>
<range> ::= <index>-<index>
<bit_specification> ::= [<index>] | [<range>]
```

In the above syntax,

- `-inst <instance>` specifies the instance to which exclusion must apply. The `<instance>` can include the wildcard characters `*`, `...`, and `?`.

Note: Special characters like `[]` in the instance names must be escaped with `\`. Else, you can use `{}` to avoid the use of `\`. For example, if the instance name is `xbus0.masters[0]`, then specify it as:

```
-inst xbus0.masters\[0\]
```

or

```
-inst {xbus0.masters[0]}
```

- `-type <type>` specifies the type to which exclusion must apply. The `<type>` can include wildcard characters `*` and `?`.

Note: When you exclude a type, all instances within the type get excluded. However, when you exclude all instances of a type, the type is not excluded automatically.

- `-metrics <metrics_type>` specifies the type of metrics to be excluded. This option should be used when you want to exclude coverage of all the items of the specified type or instance. The `<metrics_type>` can be any of the following:

- `all` for all metric types
- `code` for block, expression, and toggle coverage
- `fsm` for state, transition, and arc coverage
- `functional` for both assertion and covergroup coverage
- `block` for block coverage
- `expression` for expression coverage
- `toggle` for toggle coverage
- `state` for state coverage
- `transition` for transition coverage
- `arc` for arc coverage
- `assertion` for PSL/SVA-based assertion coverage
- `covergroup` for SystemVerilog-based covergroup coverage

Note: You can specify more than one metric type by separating the metric types with a colon (`:`). For example, to exclude block and expression, use `block:expression`.

- `<coverage_object_specification>` specifies the coverage objects to be excluded. This option must be used when you want to exclude individual coverage items of the specified instance or type. The `<coverage_object_specification>` can be any of the following:

- ❑ `-assertion <assertion_name>` to specify the assertions that must be excluded. It can include wildcard characters `*` and `?`.
- ❑ `-covergroup <covergroup_name>` to specify the covergroups that must be excluded. It can include wildcard characters `*` and `?`.
- ❑ `-coveritem <covergroup_name>. <coverpoint_name>` to specify the coverpoints that must be excluded. It can include wildcard characters `*` and `?`.
- ❑ `-coverbin <covergroup_name>. <coverpoint_name>. <coverbin_name>` to specify the coverpoint bins that must be excluded. It can include wildcard characters `*` and `?`.

- ❑ `-fsm <fsm_name>` to specify the state machines that must be excluded. It can include wildcard characters * and ?.
- ❑ `-state <fsm_name>.<state_name>` to specify the states that must be excluded. It can include wildcard characters * and ?.
- ❑ `-transition <fsm_name>.<from_state>/|.<to_state>` to specify the transitions that must be excluded. It can include wildcard characters * and ?.
- ❑ `-arc <fsm_name>.<from_state>/|.<to_state>.<arc_index>` to specify the arcs that must be excluded. The `<arc_index>` cannot include wildcard character ?, but it can include wildcard character *.
- ❑ `-toggle <signal_full_name> [[<bit_specification>] [.rise | .fall]]` to specify the signals that must be excluded. The `<signal_full_name>` can include wildcard characters * and ?. The `<bit_specification>` cannot include wildcard character ?, but it can include wildcard character *. Optional keywords `rise` and `fall` indicate if rise transition must be excluded or the fall transition must be excluded.
- ❑ `-block <index_specification>` to specify the blocks that must be excluded. The `<index_specification>` cannot include wildcard character ?, but it can include wildcard character *.
- ❑ `-expression <index_specification>` to specify the expressions that must be excluded. The `<index_specification>` cannot include wildcard character ?, but it can include wildcard character *.

- `-smart` makes the exclusion action smart, that is, any entity connected to the excluded entity will be excluded implicitly.
- `-unr` is used to exclude only the items that were marked UNR by IEV. For details on the UNR flow and how items are marked UNR by IEV, see the *IEV User Guide*. For a few examples on how to exclude UNR items, see [Excluding Unreachable Items](#) on page 221.
- `-comment <comment>` specifies the exclusion comment for the specified instance, type, or coverage item. The exclusion comment `<comment>` must be enclosed within quotes ". For example, to add the comment as "This block cannot be covered" for block 8 of instance `dtrf_recvr_core.SPI_INST`, use the following command:

```
exclude -inst dtrf_recvr_core.SPI_INST -block 8 -comment "This block cannot be covered"
```

Note: Comment cannot be more than 500 characters.

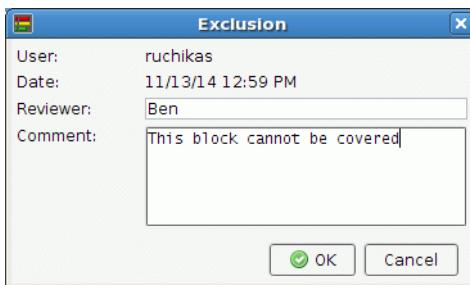
- `-reviewer <reviewer_name>` specifies the exclusion reviewer of the specified instance, type, or coverage item. For example, to specify the exclusion reviewer as `Ben` for block 8 of instance `dtrf_recvr_core.SPI_INST`, use the following command:

```
exclude -inst dtmf_recv_core.SPI_INST -block 8 -comment "This block cannot be
covered" -reviewer Ben
```

Note: If no reviewer is specified at the time of exclusion, the default value `unknown` is considered as the reviewer. In addition, the reviewer name cannot be more than 24 characters.

Note: The exclusion comments and reviewer details added and saved in IMC command-line interactive mode are available in the IMC GUI and vice versa. For example, if the above exclusions are saved in a refinement file, you can load them in GUI and also edit them. [Figure 3-26 on page 191](#) shows the *Exclusion* dialog box in IMC GUI, which shows the details added from IMC command-line.

Figure 3-26 Editing Exclusion Comments in IMC GUI



3.2.2 Excluding All Coverage Objects in an Instance or Type

The `exclude` command allows you to:

- Exclude all coverage objects in a specified instance (using the `-inst` option)
- Exclude all coverage objects in a specified type (using the `-type` option)

For example, to exclude all coverage objects in instance `dtmf_recv_core.SPI_INST`, use:

```
exclude -inst dtmf_recv_core.SPI_INST
```

To exclude all coverage objects in instance hierarchy rooted at `test.DUT.I1` including `test.DUT.I1`, use:

```
exclude -inst test.DUT.I1...
```

To exclude all coverage objects in type `dma`, use:

```
exclude -type dma
```

3.2.3 Excluding Blocks

The `exclude` command allows you to:

- Exclude all of the blocks of the specified instance or type (using the `-metrics` block option)
- Exclude specific blocks from a particular instance or type (using the `-block` option)

3.2.3.1 Excluding All of the Blocks

To exclude all of the blocks of a specific instance or type, use the `-metrics` block option of the `exclude` command, as shown below:

```
exclude [-type <type> | -inst <instance>] -metrics block
```

For example, to exclude all of the blocks of instance `dtmpf_recv_core.SPI_INST`, use:

```
exclude -inst dtmf_recv_core.SPI_INST -metrics block
```

To exclude all of the blocks of type `spi`, use:

```
exclude -type spi -metrics block
```

To exclude all the blocks in all the classes declared in the package `top`, use:

```
exclude -type top::* -metrics block
```

To exclude all the blocks in all the classes declared in module `mod1`, use:

```
exclude -type mod1_* -metrics block
```

3.2.3.2 Excluding Specific Blocks

To exclude specific blocks from coverage analysis, use the `-block` option of the `exclude` command, as shown below:

```
exclude [-type <type> | -inst <instance>] -block <index_specification>
<index_specification> ::= <index> | <range>
<range> ::= <index>-<index>
```

where `<index_specification>` specifies the report items to be excluded. It can be:

- `<index>` to specify the block number for exclusion. You can specify multiple blocks by separating them by a space character.
- `<range>` to specify a range of blocks for exclusion. The right index of the range must be greater than the left index.

Note: The *<index_specification>* should include positive integer values. In addition, *<index_specification>* cannot include wildcard character ?, but it can include wildcard character *.

For example, consider a sample Block Coverage report shown in [Figure 3-27](#) on page 193.

Figure 3-27 Block Coverage Report

Count	Block #	Stmt	Line	Kind	Origin	Source Code
0	1	0	18	ternary 1 true *	18	assign m_rcc_clk = test_mode ? ~clk : rcc_clk;
230	2	0	18	ternary 1 false *	18	assign m_rcc_clk = test_mode ? ~clk : rcc_clk;
0	3	0	19	ternary 1 true *	19	assign m_digit_clk = test_mode ? clk : digit_clk;
230	4	0	19	ternary 1 false *	19	assign m_digit_clk = test_mode ? clk : digit_clk;
0	5	0	20	ternary 1 true *	20	assign m_spi_clk = test_mode ? clk : spi_clk;
230	6	0	20	ternary 1 false *	20	assign m_spi_clk = test_mode ? clk : spi_clk;
0	7	0	21	ternary 1 true *	21	assign m_ram_clk = test_mode ? ~clk : ram_clk;
230	8	0	21	ternary 1 false *	21	assign m_ram_clk = test_mode ? ~clk : ram_clk;
0	9	0	22	ternary 1 true *	22	assign m_dsram_clk = test_mode ? ~clk : dsram_clk;
230	10	0	22	ternary 1 false *	22	assign m_dsram_clk = test_mode ? ~clk : dsram_clk;

To exclude block 1, use:

```
exclude -inst dtmf_recv_core.TEST_CONTROL_INST -block 1
```

To exclude blocks 3 and 5, use:

```
exclude -inst dtmf_recv_core.TEST_CONTROL_INST -block 3 5
```

To exclude blocks 7 to 9, use:

```
exclude -inst dtmf_recv_core.TEST_CONTROL_INST -block 7-9
```

After executing the above commands, blocks 1, 3, 5, 7, 8, and 9 are excluded from coverage analysis.

[Figure 3-28](#) on page 193 shows the report after excluding the blocks listed above.

Figure 3-28 Report After Excluding Blocks

Count	Block #	Stmt	Line	Kind	Origin	Source Code
230	2	0	18	ternary 1 false *	18	assign m_rcc_clk = test_mode ? ~clk : rcc_clk;
230	4	0	19	ternary 1 false *	19	assign m_digit_clk = test_mode ? clk : digit_clk;
230	6	0	20	ternary 1 false *	20	assign m_spi_clk = test_mode ? clk : spi_clk;
230	10	0	22	ternary 1 false *	22	assign m_dsram_clk = test_mode ? ~clk : dsram_clk;

In the above report, blocks 1, 3, 5, 7, 8, and 9 no longer show in the coverage results and the total number of blocks appear as 4, instead of 10.

To view a list of excluded blocks, use:

```
report -detail -excludes -metrics block -inst dtmf_recv_core.TEST_CONTROL_INST
```

[Figure 3-29 on page 194](#) shows the excluded blocks.

Figure 3-29 Report Showing Excluded Blocks

Number of excluded blocks: 6 Number of excluded branches: 6						
Count	Block #	Stmt	Line	Kind	Origin	Source Code
EXCL 1	0	18		ternary 1 true *	18	assign m_rcc_clk = test_mode ? ~clk : rcc_clk;
EXCL 3	0	19		ternary 1 true *	19	assign m_digit_clk = test_mode ? clk : digit_clk;
EXCL 5	0	20		ternary 1 true *	20	assign m_spi_clk = test_mode ? clk : spi_clk;
EXCL 7	0	21		ternary 1 true *	21	assign m_ram_clk = test_mode ? ~clk : ram_clk;
EXCL 8	0	21		ternary 1 false *	21	assign m_ram_clk = test_mode ? ~clk : ram_clk;
EXCL 9	0	22		ternary 1 true *	22	assign m_dsram_clk = test_mode ? ~clk : dsram_clk;

Blocks marked excluded (EXCL)

In the above report, the excluded blocks are marked as EXCL.

3.2.4 Excluding Expressions

The exclude command allows you to:

- Exclude all of the expressions of the specified instance or type (using the `-metrics expression` option)
- Exclude specific expressions from a particular instance or type (using the `-expression` option)

3.2.4.1 Excluding All of the Expressions

To exclude all of the expressions in a specific instance or type, use the `-metrics expression` option of the exclude command, as shown below:

```
exclude [-type <type> | -inst <instance>] -metrics expression
```

For example, to exclude all of the expressions in instance `dtsmf_recv_core.DECODE_INST`, use:

```
exclude -inst dtsmf_recv_core.DECODE_INST -metrics expression
```

To exclude all of the expressions in type `dma`, use:

```
exclude -type dma -metrics expression
```

3.2.4.2 Excluding Specific Expressions

To exclude specific expressions from coverage analysis, use the `-expression` option of the `exclude` command, as shown below:

```
exclude [-type <type> | -inst <instance>] -expression <index_specification>
<index_specification> ::= <index> | <range>
<range> ::= <index>-<index>
```

where `<index_specification>` specifies the report items to be excluded. It can be:

- `<index>` to specify the expression number for exclusion. You can specify multiple expressions by separating them by a space character.
- `<range>` to specify a range of expressions for exclusion. The right index of the range must be greater than the left index.

Note: The `<index_specification>` should include positive integer values. In addition, `<index_specification>` cannot include wildcard character `?`, but it can include wildcard character `*`.

Consider a sample Expression Coverage report shown in [Figure 3-30](#) on page 195.

Figure 3-30 Expression Coverage Report

```
Instance name: dtmf_recv_core.TDSP_CORE_INST.TDSP_CORE_MACH_INST
Type name: tdsp_core_mach
File name: /home/ruchikas/cov_demo_dtmf/vlog_src/tdsp_core_mach.v
Number of covered expressions: 7 of 9
Number of uncovered expressions: 2 of 9
Number of excluded expressions: 0
Number of unreachable expressions: 0

index | grade      | line | expression
-----+-----+-----+-----+
1.1  | 100.00% (3/3) | 95  | (! bus_request) || (bus_request && bus_grant)
2.1  | 33.00% (1/3) | 106 | bus_request && (! bus_grant)
3.1  | 100.00% (3/3) | 119 | bus_request && bus_grant

index: 1.1 grade: 100.00% (3/3) line: 95 source: if ((!bus_request) || (bus_request && bus_grant))
      (! bus_request) || (bus_request && bus_grant)
      <----1---->           <---2--->

index | hit | rval | <1> <2>
-----+-----+-----+-----+
1.1.1 | 1  | 1  | -  1
1.1.2 | 1  | 1  | 0  -
1.1.3 | 1  | 0  | 1  0

index: 2.1 grade: 33.00% (1/3) line: 106 source: else if (bus_request && !bus_grant)
      bus_request && (! bus_grant)
      <----1---->           <---2--->

index | hit | rval | <1> <2>
-----+-----+-----+-----+
2.1.1 | 1  | 1  | 1  0
2.1.2 | 0  | 0  | -  1
2.1.3 | 0  | 0  | 0  -
```

To exclude expression 1.1 from instance

```
dtmf_recv_core.TDSP_CORE_INST.TDSP_CORE_MACH_INST, use:
```

```
exclude -inst dtmf_recv_core.TDSP_CORE_INST.TDSP_CORE_MACH_INST -expression 1.1
```

To exclude expressions 2.1.2 and 2.1.3, use:

```
exclude -inst dtmf_recv_core.TDSP_CORE_INST.TDSP_CORE_MACH_INST -expression 2.1.2-3
```

You can view a list of excluded expressions using the following command:

```
report -detail -excludes -metrics expression -inst  
dtmf_recv_core.TDSP_CORE_INST.TDSP_CORE_MACH_INST
```

Figure 3-31 on page 196 shows the excluded expressions.

Figure 3-31 Report Showing Excluded Expressions

Expressions marked P-EXCL
(because parent was marked excluded)

Number of excluded expressions: 5			
index	grade	line	expression
1.1	EXCL	95	(! bus_request) (bus_request && bus_grant)
2.1	100.00% (1/1/2)	106	bus_request && (! bus_grant)
index: 1.1 grade: EXCL line: 95 source: if ((!bus_request) (bus_request && bus_grant)) (! bus_request) (bus_request && bus_grant) -----1-----> -----2---->			
index	hit	rval	<1> <2>
1.1.1	P-EXCL	1	- 1
1.1.2	P-EXCL	1	0 -
1.1.3	P-EXCL	0	1 0
index: 2.1 grade: 100.00% (1/1/2) line: 106 source: else if (bus_request && !bus_grant) bus_request && (! bus_grant) -----1-----> -----2---->			
index	hit	rval	<1> <2>
2.1.2	EXCL	0	- 1
2.1.3	EXCL	0	0 -

Expressions marked excluded (EXCL)

In the above report:

- Expressions 2.1.2 and 2.1.3 show as EXCL (to indicate excluded).
- Expressions 1.1.1, 1.1.2, and 1.1.3 show as P-EXCL (to indicate that these items were excluded because the parent 1.1 was marked excluded).

Important

Items marked P-EXCL are excluded because the parent was excluded. You cannot *un-exclude* these items individually. To *un-exclude* these items, you must *un-exclude* the parent.

Consider another example where exclusion is made from type.

```
exclude -type alu_32 -expression 1.1
```

In this case, expression 1.1 is excluded from type alu_32.

Now generate a type-based report to see excluded expressions using the following command:

```
report -detail -excludes -metrics expression -type alu_32
```

[Figure 3-32 on page 197](#) shows the type-based report after excluding above expression.

Figure 3-32 Expression Coverage Type-Based Report

Expression marked EXCL because it was excluded explicitly

```
Type name: alu_32
File name: /home/ruchikas/cov_demo_dtmf/vlog_src/alu_32.v
Number of excluded expressions: 3

index | grade      | line   | expression
-----+-----+-----+-----+
1.1  | EXCL       | 55     | (ovm && ovf)

index: 1.1 grade: EXCL line: 55 source: assign #1 sat_prod = (ovm && ovf) ? (prod['HMSB']) ? PSAT : NSAT : prod :
(ovm && ovf)
<1>   <2>

index | hit | <1> <2>
-----+-----+-----+
1.1.1 | P-EXCL | 0   -
1.1.2 | P-EXCL | -   0
1.1.3 | P-EXCL | 1   1
```

Expressions marked P-EXCL (because parent was marked excluded)

Now, consider generating an instance-based report for the same entity:

```
report -detail -excludes -metrics expression -inst
dtmf_recvrx_core.TDSP_CORE_INST.ALU_32_INST
```

[Figure 3-33 on page 198](#) shows the excluded expressions.

Figure 3-33 Report Showing Excluded Expressions

Expression marked T-EXCL (because it was excluded from type)

```

Instance name: dtmf_recv_core.TDSP_CORE_INST.ALU_32_INST
Type name: alu_32
File name: /home/ruchikas/cov_demo_dtmf/vlog_src/alu_32.v
Number of excluded expressions: 3

index | grade      | line | expression
-----|-----|-----|-----
1.1  | T-EXCL    | 55   | (ovm && ovf)

index: 1.1 grade: T-EXCL line: 55 source: assign #1 sat_prod = (ovm && ovf) ? (prod['HMSB]) ? PSAT : NSAT : prod
(ovm && ovf)
<1>    <2>

index | hit | <1> <2>
-----|-----|-----'|&&
1.1.1 | P-EXCL | 0  -
1.1.2 | P-EXCL | -  0
1.1.3 | P-EXCL | 1  1

```

Expressions marked P-EXCL (because parent was marked excluded)

In the above report, notice that in the case of instance-based report, expression 1.1 is marked as T-EXCL. This is because the expression was excluded from the type. T-EXCL is valid only for instance-based reports for the expressions that are excluded from types.

3.2.5 Excluding Toggle Signals

The `exclude` command allows you to:

- Exclude all of the signals of the specified instance or type (using the `-metrics toggle` option)
- Exclude specific signals from a particular instance or type (using the `-toggle` option)

3.2.5.1 Excluding All Signals

To exclude all of the signals of a specific instance or type, use the `-metrics toggle` option of the `exclude` command, as shown below:

```
exclude [-type <type> | -inst <instance>] -metrics toggle
```

For example, to exclude all of the signals of instance `dtmf_recv_core.AR_B_INST`, use:

```
exclude -inst dtmf_recv_core.AR_B_INST -metrics toggle
```

3.2.5.2 Excluding Specific Signals

To exclude the specific signals from coverage analysis, use the `-toggle` option of the `exclude` command, as shown below:

```
exclude [-type <type> | -inst <instance>]
-toggle <signal_full_name>[<bit_specification>] [.rise | .fall]
<bit_specification> ::= [<index>] | [<range>]
<range> ::= <index>-<index>
```

where

- `<signal_full_name>` is the name of the signal that should be excluded.
- `<bit_specification>` specifies the signal bits for exclusion. Signal bits can be specified as any of the following:
 - Single bits using `<index>`
 - Range of bits using `<range>`
- Note:** The `<bit_specification>` should include positive integer values. In addition, `<bit_specification>` cannot include wildcard character `?`, but it can include wildcard character `*`.
- `[.rise | .fall]` specifies if the rise transition or the fall transition of the specified signal must be excluded.

For example, consider the sample Toggle Coverage report shown in [Figure 3-34](#) on page 200.

Figure 3-34 Toggle Coverage Report

```
Instance name: dtmf_recv_core.AR_B_INST
Type name: arb
File name: /home/ruchikas/cov_demo_dtmf/vlog_src/arb.v
Number of covered signal bits: 12 of 15
Number of uncovered signal bits: 3 of 15
Number of excluded signal bits: 0
Number of unreachable signal bits: 0
Number of signal bits partially toggled(rise): 0 of 15
Number of signal bits partially toggled(fall): 0 of 15

Hit(Full) Hit(Rise) Hit(Fall) Signal
-----
1 1 1 reset
1 1 1 clk
1 1 1 dma_breq
1 1 1 tdsp_breq
1 1 1 dma_grant
1 1 1 tdsp_grant
0 0 0 scan_in0
0 0 0 scan_en
0 0 0 scan_out0
1 1 1 next_state[2]
1 1 1 next_state[1]
1 1 1 next_state[0]
1 1 1 present_state[2]
1 1 1 present_state[1]
1 1 1 present_state[0]
```

Examples -- Excluding complete signals

To exclude the signal bits `present_state[0]` and `present_state[1]`, use:

```
exclude -inst dtmf_recv_core.AR_B_INST -toggle {present_state[0-1]}
```

Note: If you do not specify the signal bits with the signal name, all bits are excluded.

To exclude the signal `dma_breq`, use:

```
exclude -inst dtmf_recv_core.AR_B_INST -toggle dma_breq
```

Examples -- Excluding partial signals (rise or fall transitions)

To exclude only the rise transition of signal bit `present_state[2]`, use:

```
exclude -inst dtmf_recv_core.AR_B_INST -toggle present_state\[2.rise\]
```

To exclude only the rise transition of all the signal bits of `next_state`, use:

```
exclude -inst dtmf_recv_core.AR_B_INST -toggle next_state\[*.rise\]
```

To exclude only the fall transition of signal `reset` (signal `reset` is single bit signal), use:

```
exclude -inst dtmf_recv_core.AR_B_INST -toggle reset\[*.fall\]
```

Note: In the `exclude` command, no mention of bits with `\[.rise\]` or `\[.fall\]` will exclude all bit(s) in that signal regardless of signal being a vector of bits or a single bit.

Figure 3-35 on page 201 shows the report after executing above commands.

Figure 3-35 Report After Excluding Signals

Number of covered signal bits: 9 of 12 Number of uncovered signal bits: 3 of 12 Number of excluded signal bits: 3 Number of unreachable signal bits: 0 Number of signal bits partially toggled(rise): 0 of 8 Number of signal bits partially toggled(fall): 0 of 11			
Hit(Full)	Hit(Rise)	Hit(Fall)	Signal
1	1	EXCL	reset
1	1	1	clk
P-EXCL	P-EXCL	P-EXCL	dma_breq
1	1	1	tdsp_breq
1	1	1	dma_grant
1	1	1	tdsp_grant
0	0	0	scan_in0
0	0	0	scan_en
0	0	0	scan_out0
1	EXCL	1	next_state[2]
1	EXCL	1	next_state[1]
1	EXCL	1	next_state[0]
1	EXCL	1	present_state[2]
EXCL	EXCL	EXCL	present_state[1]
EXCL	EXCL	EXCL	present_state[0]

In the above report:

- For signal `reset`, `EXCL` is shown only in the `Hit (Fall)` column. This is because only the fall transition was excluded for it.
- For signal `dma_breq`, `P-EXCL` is shown in all the columns because the complete signal was excluded and `P-EXCL` indicates that the parent was marked excluded.
- For signals bits `next_state[0]`, `next_state[1]`, and `next_state[2]` only the `Hit (Rise)` column shows `EXCL`. This is because for these signal bits only the rise transition was marked excluded.
- For signal bit `present_state[2]`, `EXCL` is shown only in the `Hit (Rise)` column. This is because only the rise transition was excluded for it.
- For signal bits `present_state[0]` and `present_state[1]`, all columns show as `EXCL` to indicate complete exclusion.

Based on above exclusions, the report header is also updated to show the modified number of covered toggles, uncovered toggles, excluded toggles, signal bits partially toggled (rise), and signal bits partially toggled (fall).

Example -- Excluding enum toggles

Consider a sample Toggle Coverage report shown in [Figure 3-36](#) on page 202.

Figure 3-36 Toggle Coverage Report

Hit(Full)	Hit(Rise)	Hit(Fall)	Signal
1	1	1	a[3]
1	1	1	a[2]
0	0	0	a[1]
0	1	0	a[0]
1	1	1	d[3]
1	1	1	d[2]
1	1	1	d[1]
1	1	1	d[0]
1	1	1	clk
0	0	0	my_bit[1]
0	0	0	my_bit[0]
0	0	0	my_logic[1]
0	0	0	my_logic[0]
Hit	Value	Enum	
0	green	color	
1	red		
1	yellow		
0	blue		

To exclude enum toggle value `yellow` of enum toggle `color`, use:

```
exclude -inst top -toggle color\[yellow\]
```

[Figure 3-37](#) on page 202 shows the report after executing above command.

Figure 3-37 Report After Excluding Enum Toggle

Hit(Full)	Hit(Rise)	Hit(Fall)	Signal
1	1	1	a[3]
1	1	1	a[2]
0	0	0	a[1]
0	1	0	a[0]
1	1	1	d[3]
1	1	1	d[2]
1	1	1	d[1]
1	1	1	d[0]
1	1	1	clk
0	0	0	my_bit[1]
0	0	0	my_bit[0]
0	0	0	my_logic[1]
0	0	0	my_logic[0]
Hit	Value	Enum	
0	green	color	
1	red		
EXCL	yellow		
0	blue		

Enum value excluded →

To exclude the complete enum toggle `color`, use:

```
exclude -inst top -toggle color
```

The above command will exclude all the values of enumerated toggle color.

In case an enum is defined in a struct, then enclose the enum information within { " " } or \". For example, if the enum `addr` is defined in struct `port`, then use the following command to exclude the complete enum:

```
imc> exclude -inst big_inst.si -toggle {"port.addr"}
```

or

```
imc> exclude -inst big_inst.si -toggle \\"port.addr\\"
```

In case you want to exclude a particular value of enum defined within a struct (for example, value `INIT` of enum `addr` defined in struct `port`), then use the following command:

```
imc> exclude -inst big_inst.si -toggle \\"port.addr\\\"[INIT\\]
```

To exclude a particular value of enum defined in a nested struct (for example, value `INIT` of enum `high` defined in nested struct `port.incr`), use:

```
imc> exclude -inst big_inst.si -toggle \\"port.incr.high\\\"[INIT\\]
```

Example -- Exclusion of toggle array bits in case of Multi-Dimensional Arrays

Consider a sample Toggle Coverage report shown in [Figure 3-38](#) on page 204.

Figure 3-38 Toggle Coverage Report

Hit(Full)	Hit(Rise)	Hit(Fall)	Signal
0	0	0	a_drv[3][4][0]
0	0	0	a_drv[3][4][1]
0	0	0	a_drv[3][4][2]
0	0	0	a_drv[3][4][3]
0	0	0	a_drv[3][4][4]
0	0	0	a_drv[3][4][5]
0	0	0	a_drv[3][3][0]
0	0	0	a_drv[3][3][1]
0	0	0	a_drv[3][3][2]
0	0	0	a_drv[3][3][3]
0	0	0	a_drv[3][3][4]
0	0	0	a_drv[3][3][5]
0	0	0	a_drv[3][2][0]
0	0	0	a_drv[3][2][1]
0	0	0	a_drv[3][2][2]
0	0	0	a_drv[3][2][3]
0	0	0	a_drv[3][2][4]
0	0	0	a_drv[3][2][5]
0	0	0	a_drv[3][1][0]
0	0	0	a_drv[3][1][1]
0	0	0	a_drv[3][1][2]
0	0	0	a_drv[3][1][3]
0	0	0	a_drv[3][1][4]
0	0	0	a_drv[3][1][5]
0	0	0	a_drv[3][0][0]

To exclude all the toggle bits of multi-dimensional array `a_drv` where first array index is 3, use:

```
exclude -inst top -toggle a_drv\[3\]
```

To exclude all toggle bits of multi-dimensional array `a_drv` where first two array indices are `[1][0]`, use:

```
exclude -inst top -toggle a_drv\[1\]\[0\]
```

To exclude all toggle bits of multi-dimensional array `a_r` in all instances where first two array indices are `[1][1]`, use:

```
exclude -inst *... -toggle a_r\[1\]\[1\]\[*\]
```

To exclude the rise transition of the multi-dimensional array `a_r[1][2][1]`, use:

```
exclude -inst *... -toggle a_r\[1\]\[2\]\[1.rise\]
```

Note: To exclude all blocks, expressions, and signals of a specific instance or type, use the `exclude` command as:

```
exclude [-type <type> | -inst <instance>] -metrics code
```

For example, to exclude all blocks, expressions, and signals of instance `dtrmf_recv_core.AR_B_INST`, use:

```
exclude -inst dtrmf_recv_core.AR_B_INST -metrics code
```

3.2.6 Excluding FSMs

The `exclude` command allows you to:

- Exclude all of the FSMs of the specified instance or type (using the `-metrics fsm` option)
- Exclude a specific FSM of the specified instance or type (using the `-fsm` option)
- Exclude all states of all of the FSMs in the specified instance or type (using the `-metrics state` option)
- Exclude specific states of an FSM (using the `-state` option)
- Exclude all transitions of all of the FSMs in the specified instance or type (using the `-metrics transition` option)
- Exclude specific transitions of an FSM (using the `-transition` option)
- Exclude all arcs of all of the FSMs in the specified instance or type (using the `-metrics arc` option)
- Exclude specific arcs of an FSM (using the `-arc` option)

3.2.6.1 Excluding All FSMs

To exclude all of the state machines of the specified instance or type, use the `-metrics fsm` option of the `exclude` command, as shown below:

```
exclude [-type <type> | -inst <instance>] -metrics fsm
```

For example, to exclude all state machines of instance `dtmpf_recv_core.SPI_INST`, use:

```
exclude -inst dtmf_recv_core.SPI_INST -metrics fsm
```

With the above command, all state machines of instance `dtmpf_recv_core.SPI_INST` are excluded from coverage analysis.

3.2.6.2 Excluding Specific FSMs

To exclude a specific FSM from coverage analysis, use the `-fsm` option of the `exclude` command, as shown below:

```
exclude [-type <type> | -inst <instance>] -fsm <fsm_name>
```

For example, to exclude FSM `present_state` from coverage analysis, use:

```
exclude -inst dtmf_recv_core.SPI_INST -fsm present_state
```

With the above command, all of the states, transitions, and arc available in FSM `present_state` are excluded from coverage analysis.

3.2.6.3 Excluding All States of an FSM

To exclude all states of all of the FSMs of a specified instance or type, use the `-metrics state` option of the `exclude` command, as shown below:

```
exclude [-type <type> | -inst <instance>] -metrics state
```

For example, to exclude all states of all of the FSMs in instance `dtrmf_recv_core.SPI_INST`, use:

```
exclude -inst dtrmf_recv_core.SPI_INST -metrics state
```

With the above command, all states of all of the FSMs of instance `dtrmf_recv_core.SPI_INST` are excluded from coverage analysis.

3.2.6.4 Excluding Specific States

To exclude specific states of a state machine, use the `-state` option of the `exclude` command, as shown below:

```
exclude [-type <type> | -inst <instance>] -fsm <fsm_name>.<state_name>
```

where

- `<fsm_name>` is the name of the state machine from which you want to exclude states.
- `<state_name>` is the name of the state that you want to exclude.

Note: When excluding a reset state or transition, `_RST` should be appended to the state name. For example, if you want to exclude a reset state named `State_1`, then the command line should specify the reset state as `State_1_RST`.

Consider a sample State Coverage report shown in [Figure 3-39](#) on page 207.

Figure 3-39 State Coverage Report

```
Instance name: dtmf_recv_core.SPI_INST
Type name: spi
File name: /home/ruchikas/cov_demo_dtmf/vlog_src/spi.v.gz
State register: present_state
Number of covered states: 6 of 6
Number of uncovered states: 0 of 6
Number of excluded states: 0

State Coverage:
=====
State Encoding Visits
-----
State_1 001 45904
State_0 000 17
State_2 010 833
State_3 011 6375
State_7 111 17
```

To exclude the state State_1, use:

```
exclude -inst dtmf_recv_core.SPI_INST -state present_state.State_1
```

With the above command, State_1 is excluded from coverage analysis.

[Figure 3-40 on page 207](#) shows the report after excluding State_1.

Figure 3-40 Report After Excluding State_1

```
State register: present_state
Number of covered states: 5 of 5
Number of uncovered states: 0 of 5
Number of excluded states: 1

State Coverage:
=====
State Encoding Visits
-----
State_1 001 EXCL
State_0 000 17
State_2 010 833
State_3 011 6375
State_7 111 17
```

State_1 marked Excluded (EXCL)

In the above report, State_1 is shown as excluded and the total number of states appear as 5 instead of 6.

You can also exclude multiple states using a single exclude command. For example, to exclude State_0 and State_3, use:

```
exclude -inst dtmf_recv_core.SPI_INST -state present_state.State_0
present_state.State_3
```

Note: To exclude a reset state or transition, _RST should be appended to the state name. For example, if you want to exclude a reset state named State_1, then the command line should specify the reset state as State_1_RST, as shown below:

```
exclude -inst dtmf_recv_core.SPI_INST -state present_state.State_1_RST
```

Note: Currently, the `-excludes` option does not print the excluded states. It only prints the number of excluded states in the report header. The actual excluded state is not printed in the report body. You can view the excluded states using the `-all` option of the report `-detail` command.

3.2.6.5 Excluding All Transitions of all of the FSM

To exclude all transitions of all of the FSMs of a specified instance or type, use the `-metrics transition` option of the `exclude` command, as shown below:

```
exclude [-type <type> | -inst <instance>] -metrics transition
```

For example, to exclude all transitions of all of the FSMs in instance `dtmf_recv_core.SPI_INST`, use:

```
exclude -inst dtmf_recv_core.SPI_INST -metrics transition
```

With the above command, all transitions of all of the FSMs of instance `dtmf_recv_core.SPI_INST` are excluded from coverage analysis.

3.2.6.6 Excluding Specific Transitions

To exclude specific transitions of a state machine, use the `-transition` option of the `exclude` command, as shown below:

```
exclude [-type <type> | -inst <instance>]  
-transition <fsm_name>.<from_state>/|.<to_state>
```

where

- `<fsm_name>` is the name of the state machine from which you want to exclude transitions.
- `<from_state>` is the name of the present state that indicates the beginning of a transition.
- `/ | .` indicates the separator between the `<from_state>` and the `<to_state>`. Separator can be `"/` or `".`.
- `<to_state>` is the name of the next state that indicates the end state of a transition.

Note: When excluding a reset state or transition, `_RST` should be appended to the state name. For example, if you want to exclude a reset state named `State_1`, then the command line should specify the reset state as `State_1_RST`.

Consider a sample Transition Coverage report shown in [Figure 3-41](#) on page 209.

Figure 3-41 Transition Coverage Report

State register: present_state		
Number of covered transitions: 9 of 14		
Number of uncovered transitions: 5 of 14		
Number of excluded transitions: 0		
Transition Coverage:		
=====		
P-State	N-State	Visits

State_1	State_0	17
	State_1	45886
State_0	State_2	17
	State_1	0
State_2	State_2	816
	State_3	17
State_3	State_7	17
	State_3	9316
State_7	State_1	17

To exclude the transition State_1 -> State_0, use:

```
exclude -inst dtmf_recv_core.SPI_INST -transition present_state.State_1.State_0
```

With the above command, transition State_1 -> State_0 is excluded from coverage results.

[Figure 3-42](#) on page 209 shows the report after excluding the transition State_1 -> State_0.

Figure 3-42 Report After Excluding State_1 -> State_0

State register: present_state		
Number of covered transitions: 8 of 13		
Number of uncovered transitions: 5 of 13		
Number of excluded transitions: 1		
Transition Coverage:		
=====		
P-State	N-State	Visits

State_1	State_0	EXCL
	State_1	45886
State_0	State_2	17
	State_1	0
State_2	State_2	816
	State_3	17
State_3	State_7	17
	State_3	9316
State_7	State_1	17

Total number of transitions now show 13 instead of 14

Transition State_1 -> State_0 no longer appears in the report.

Note: If arc scoring is enabled, arcs are reported as input conditions under which a transition takes place. Arcs associated with a transition are automatically excluded when you exclude a transition. In addition, you can exclude arcs individually in both batch mode as well as the GUI mode.

You can view the excluded transitions, using the `-excludes` option of the `report -detail` command.

3.2.6.7 Excluding All Arcs of all of the FSM

To exclude all arcs of all of the FSMs of a specified instance or type, use the `-metrics arc` option of the `exclude` command, as shown below:

```
exclude [-type <type> | -inst <instance>] -metrics arc
```

For example, to exclude all arcs of all of the FSMs in instance `dtrmf_recv_core.SPI_INST`, use:

```
exclude -inst dtrmf_recv_core.SPI_INST -metrics arc
```

With the above command, all arcs of all of the FSMs of instance `dtrmf_recv_core.SPI_INST` are excluded from coverage analysis.

3.2.6.8 Excluding Specific Arcs

To exclude specific arcs of a state machine, use the `-arc` option of the `exclude` command, as shown below:

```
exclude [-type <type> | -inst <instance>]  
-arc <fsm_name>.<from_state>.<to_state>.<arc_index>
```

where

- `<fsm_name>` is the name of the state machine from which you want to exclude arcs.
- `<from_state>` is the name of the present state that indicates the beginning of a transition.
- `/ | .` indicates the separator between the `<from_state>` and the `<to_state>`. Separator can be "/" or ".".
- `<to_state>` is the name of the next state that indicates the end state of a transition.
- `<arc_index>` specifies the arcs to be excluded. You can specify the `arc_index` as:
 - `<index>` to specify the arc number for exclusion. You can specify multiple arcs by separating them by a space character.
 - `<range>` to specify a range of arcs for exclusion. The right index of the range must be greater than the left index.

Note: The arc index starts from number 0. In addition, `<arc_index>` cannot include wildcard character ?, but it can include wildcard character *.

For example, consider a sample Arc Coverage report shown in [Figure 3-43](#) on page 211.

Figure 3-43 Arc Coverage Report

Instance name: dtmf_recv_core.SPI_INST			
Type name: spi			
State register: present_state			
Number of covered arcs: 10 of 18			
Number of uncovered arcs: 8 of 18			
Number of excluded arcs: 0			
Arc Coverage:			
=====			
Inputs = (spi_fs, spi_clk, not_full, full)			
P-State	N-State	Inputs	Visits

State_1	State_0	10--	17
	State_1	-1--	0
		0---	45886
State_0	State_2	10--	17
	State_1	-1--	0
		0---	0
State_2	State_2	-01-	816
	State_3	-1--	17
		--0-	0
State_3	State_7	-0-1	17
	State_3	-1--	3383
		---0	5933
State_7	State_1	----	17
Reset Transitions:			
=====			
P-State	Reset State	Visits	

State_1	State_1	1	
State_0	State_1	0	
State_2	State_1	0	
State_3	State_1	0	
State_7	State_1	0	

Annotations on the left side of the table:

- State2 --> State3
- Arc Index 0
- Arc Index 1

Arrows point from the annotations to the table cells for State_2 and State_3.

The Input column lists all possible arcs for each transition. In the above report, transition State_2 -> State_3 happens under two arc conditions, -1-- and --0-, respectively.

To exclude the first arc condition of State_2 -> State_3, use:

```
exclude -inst dtmf_recv_core.SPI_INST -arc present_state.State_2.State_3.0
```

With the above command, the first arc of State_2 -> State_3 is excluded from coverage results.

[Figure 3-44](#) on page 212 shows the report after excluding the specified arc.

Figure 3-44 Report After Excluding the Arc

Instance name: dtmf_recv_core.SPI_INST
 Type name: spi
 State register: present state
 Number of covered arcs: 9 of 17
 Number of uncovered arcs: 8 of 17
 Number of excluded arcs: 1

Arc Coverage:

P-State	N-State	Inputs	Visits
State_1	State_0	10--	17
	State_1	-1--	0
		0---	45886
State_0	State_2	10--	17
	State_1	-1--	0
		0---	0
State_2	State_2	-01-	816
	State_3	-1--	EXCL
		--0-	0
State_3	State_7	-0-1	17
	State_3	-1--	3383
		---0	5933
State_7	State_1	----	17

Reset Transitions:

P-State	Reset State	Visits
State_1	State_1	1
State_0	State_1	0
State_2	State_1	0
State_3	State_1	0
State_7	State_1	0

The first arc condition of State_2 → State_3 is now excluded.

In case you wanted to exclude both the arcs of State_2 → State_3, you could have used any of the following:

- Specifying multiple arcs (Separating arcs using a space character)

```
exclude -inst dtmf_recv_core.SPI_INST -arc present_state.State_2.State_3.0
       present_state.State_2.State_3.1
```

- Specifying multiple arcs (Specifying arcs as a range)

```
exclude -inst dtmf_recv_core.SPI_INST -arc present_state.State_2.State_3.0-1
```

- Specifying all of the arcs (Using wildcard *)

```
exclude -inst dtmf_recv_core.SPI_INST -arc present_state.State_2.State_3.*
```

3.2.7 Excluding Assertions

The exclude command allows you to:

- Exclude all of the assertions of the specified instance or type (using the `-metrics assertion` option)
- Exclude specific assertions from a particular instance or type (using the `-assertion` option)

3.2.7.1 Excluding All of the Assertions

To exclude all of the assertions of a specific instance or type, use the `-metrics assertion` option of the `exclude` command, as shown below:

```
exclude [-type <type> | -inst <instance>] -metrics assertion
```

For example, to exclude all of the assertions in the instance `dtsmf_recv_core.SPI_INST`, use:

```
exclude -inst dtmf_recv_core.SPI_INST -metrics assertion
```

3.2.7.2 Excluding Specific Assertions

To exclude specific assertions from coverage analysis, use the `-assertion` option of the `exclude` command, as shown below:

```
exclude [-type <type> | -inst <instance>] -assertion <assertion_name>
```

Consider a sample Assertion Coverage report shown in [Figure 3-45](#) on page 213.

Figure 3-45 Assertion Coverage Report

Assertion Coverage Report					
Instance name: dtmf_recv_core.SPI_INST					
Type name: spi					
File name: /home/ruchikas/cov_demo_dtsmf/vlog_src/spi.v					
Number of covered assertions: 6 of 9					
Number of uncovered assertions: 3 of 9					
Finished	Failed	Assertion	Line	Source	Code
0	0	__cover4_spi_interruptedframe	246	// ps1	cover spi_interruptedframe;
17	0	__cover3_spi_completemframe	244	// ps1	cover spi_completemframe;
425	0	__cover2_spi_oneframe	242	// ps1	cover spi_oneframe;
4	0	__cover1_spi_reset	240	// ps1	cover spi_reset;
17	408	spi_fs_once	195	// ps1	assume spi_fs_once;
0	425	spi_fs_start	202	// ps1	assume spi_fs_start;
136	0	spi_nooverflow	213	// ps1	assume spi_nooverflow;
0	17	spi_count8	225	// ps1	assert spi_count8;
102	0	spi_datahelduntilread	231	// ps1	assert spi_datahelduntilread;

To exclude assertion `spi_fs_start` from coverage results, use:

```
exclude -inst dtmf_recv_core.SPI_INST -assertion spi_fs_start
```

[Figure 3-46 on page 214](#) shows the report after executing the above command.

Figure 3-46 Report After Excluding spi_fs_start

File name: /home/ruchikas/cov_demo_dtmf/vlog_src/spi.v					
Number of covered assertions: 6 of 8			Number of uncovered assertions: 2 of 8		
Finished	Failed	Assertion	Line	Source	Code
0	0	__cover4_spi_interruptedframe	246	// ps1	cover spi_interruptedframe;
17	0	__cover3_spi_completemframe	244	// ps1	cover spi_completemframe;
425	0	__cover2_spi_oneframe	242	// ps1	cover spi_oneframe;
4	0	__cover1_spi_reset	240	// ps1	cover spi_reset;
17	408	spi_fs_once	195	// ps1	assume spi_fs_once;
136	0	spi_nooverflow	213	// ps1	assume spi_nooverflow;
0	17	spi_count8	225	// ps1	assert spi_count8;
102	0	spi_datahelduntilread	231	// ps1	assert spi_datahelduntilread;

The results now show total assertions as 8 instead of 9. The assertion `spi_fs_start` is no longer listed in the report.

To view a list of excluded assertions, use:

```
report -detail -excludes -metrics assertion -inst dtmf_recv_core.SPI_INST
```

[Figure 3-47 on page 214](#) shows the excluded assertions.

Figure 3-47 Excluded Assertions

Number of excluded assertions: 1					
Finished Failed Assertion			Line	Source	Code
EXCL	EXCL	spi_fs_start	202	// ps1	assume spi_fs_start;

The assertion `spi_fs_start` shows as excluded (EXCL).

Note: You can specify multiple assertions on the command line by separating them by space character. For example, to exclude `spi_fs_once` and `spi_count8`, use:

```
exclude -inst dtmf_recv_core.SPI_INST -assertion spi_fs_once spi_count8
```

3.2.8 Excluding Covergroup Items

The `exclude` command allows you to:

- Exclude all of the covergroups of the specified instance or type (using the `-metrics covergroup` option)

- Exclude specific covergroups (using the `-covergroup` option)
- Exclude specific coverpoints (using the `-coveritem` option)
- Exclude specific coverpoint bins (using the `-coverbin` option)

Important Considerations While Excluding Covergroup Items

When specifying the covergroup, coverpoint, or coverbin details in the `exclude` command, remember that:

- Special characters (like `[]`) or hierarchy delimiters (like `::` or `:`) must be escaped with `\`.

Example 1: To exclude coverbin `vector_low[20]` of coverpoint `A2`, use:

```
exclude -inst dtmf_recv_core.TDSP_CORE_INST -coverbin cg.A2.vector_low\[20\]
```

Example 2: To exclude coverbin `auto[NOP],size[1]` of coverpoint `trans_dirXsize`, of covergroup `co_trans`, which is embedded in class `xbus_master_monitor`, use:

```
exclude -inst xbus_tb_top -coverbin xbus_master_monitor\::\:co_trans.trans_dirXsize.auto\[NOP\],size\[1\]
```

- To avoid the use of `\`, you can enclose the complete bin information within `{ }` and enclose the covergroup name, the coverpoint name, and the bin name within quotes `" "`.

For example, the above examples can be modified as follows to avoid the use of `\`.

Example 1:

```
exclude -inst dtmf_recv_core.TDSP_CORE_INST -coverbin {"cg"."A2"."vector_low[20]"}
```

Example 2:

```
exclude -inst xbus_tb_top -coverbin {"xbus_master_monitor":\:co_trans"."trans_dirXsize"."auto[NOP],size[1]"}
```

3.2.8.1 Excluding All Covergroups

To exclude all of the covergroups of the specified instance or type, use the `-metrics covergroup` option of the `exclude` command, as shown below:

```
exclude [-type <type> | -inst <instance>] -metrics covergroup
```

For example, to exclude all covergroups of instance

`dtmf_recv_core.TDSP_CORE_INST`, use:

```
exclude -inst dtmf_recv_core.TDSP_CORE_INST -metrics covergroup
```

With the above command, all covergroups of instance

`dtmf_recv_core.TDSP_CORE_INST` are excluded from coverage analysis.

3.2.8.2 Excluding Specific Covergroups

To exclude specific covergroups from coverage analysis, use the `-covergroup` option of the `exclude` command, as shown below:

```
exclude [-type <type> | -inst <instance>] -covergroup <covergroup_name>
```

For example, to exclude covergroup `cg` of instance `dtmpf_recv_core.TDSP_CORE_INST` from coverage analysis, use:

```
exclude -inst dtmf_recv_core.TDSP_CORE_INST -covergroup cg
```

With the above command, covergroup `cg` is completely excluded from coverage results. As a result, all of the coverpoints, coverbins, and crosses available in the covergroup `cg` are also excluded from coverage analysis.

Note: A dot (.) is not considered as a special character when it used in the context of hierarchical paths. However, when a dot is used in covergroup names or embedded covergroups, it is considered as a special character. In such cases, a dot must be escaped with a \ or the covergroup name must be specified within ". ". For example, to exclude covergroup named `.cg` embedded within class `mc1`, use:

```
exclude -inst top -covergroup {"mc1.cg"}
```

or

```
exclude -inst top -covergroup {mc1\.cg}
```

If a covergroup is embedded within a class, then the covergroup name must include the complete hierarchical path as:

```
<class_name>::<covergroup_name>
```

For example, to exclude covergroup `CG`, which is embedded in class `MyClassType`, use the following command:

```
exclude -type test -covergroup {"MyClassType::CG"}
```

Or

```
exclude -type test -covergroup MyClassType\:::CG
```

3.2.8.3 Excluding Coveritems (Coverpoints)

To exclude specific coverpoints from coverage analysis, use the `-coveritem` option of the `exclude` command, as shown below:

```
exclude [-type <type> | -inst <instance>]  
-coveritem <covergroup_name>.<coverpoint_name>
```

Consider a sample Covergroup Coverage report of instance named `dtmpf_recv_core.TDSP_CORE_INST` shown in [Figure 3-48](#) on page 217.

Figure 3-48 Covergroup Coverage Report

Name	Average, Covered Grade	Line	Source Code
cginst1	50%, 18% (10/56)	187	covergroup cg @ (negedge clk);
-A	100% (2/2)	191	A: coverpoint TDSP_CORE_GLUE_INST.opb
cginst1.A.scalar_min	100% (358/3)	196	bins scalar_min = {0};
cginst1.A.scalar_max	100% (36/3)	197	bins scalar_max = {10};
cginst1.A2	28% (4/14)	201	A2: coverpoint TDSP_CORE_GLUE_INST.opb
-vector_low[18]	100% (42/40)	206	bins vector_low[] = {[18:20]};
-vector_low[19]	100% (45/40)	206	bins vector_low[] = {[18:20]};
-vector_low[20]	100% (42/40)	206	bins vector_low[] = {[18:20]};
-vector_med[20]	100% (42/40)	207	bins vector_med[] = {[20:30]};
-vector_med[21]	0% (36/40)	207	bins vector_med[] = {[20:30]};
-vector_med[22]	0% (36/40)	207	bins vector_med[] = {[20:30]};
-vector_med[23]	0% (15/40)	207	bins vector_med[] = {[20:30]};
-vector_med[24]	0% (0/40)	207	bins vector_med[] = {[20:30]};
-vector_med[25]	0% (6/40)	207	bins vector_med[] = {[20:30]};
-vector_med[26]	0% (0/40)	207	bins vector_med[] = {[20:30]};
-vector_med[27]	0% (0/40)	207	bins vector_med[] = {[20:30]};
-vector_med[28]	0% (0/40)	207	bins vector_med[] = {[20:30]};
-vector_med[29]	0% (0/40)	207	bins vector_med[] = {[20:30]};
-vector_med[30]	0% (0/40)	207	bins vector_med[] = {[20:30]};
cginst1.B	20% (2/10)	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
-auto[0:429496728]	100% (17314/1)	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
-auto[429496729:858993457]	100% (3/1)	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
-auto[858993458:1288490186]	0% (0/1)	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
-auto[1288490187:1717986915]	0% (0/1)	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
-auto[1717986916:2147483644]	0% (0/1)	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
-auto[2147483645:2576980373]	0% (0/1)	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
-auto[2576980374:3006477102]	0% (0/1)	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
-auto[3006477103:3435973831]	0% (0/1)	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
-auto[3435973832:3865470560]	0% (0/1)	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
-auto[3865470561:4294967295]	0% (0/1)	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
cginst1.B2	100% (2/2)	216	B2: coverpoint TDSP_CORE_GLUE_INST.opb
-opbmin	100% (565/1)	220	bins opbmin = {0};
-opbmax	100% (9/1)	221	bins opbmax = {60};
cginst1.AXA2	0% (0/28)	211	AXA2: cross A, A2;
-scalar_min.vector_low[18]	0% (0/1)	211	AXA2: cross A, A2;
-scalar_min.vector_low[19]	0% (0/1)	211	AXA2: cross A, A2;
-scalar_min.vector_low[20]	0% (0/1)	211	AXA2: cross A, A2;

Note: In the report, the grade is showing no decimal places because the value of float_precision configurable item is set to 0. By default, the value is set to 2 and you would see 2 decimal places in the attributes that show values as decimals. For more details, see [Setting Values for Configurable Items](#) on page 254.

To exclude coverpoints B and B2 from coverage results, use:

```
exclude -inst dtmf_recv_core.TDSP_CORE_INST -coveritem cginst1.B cginst1.B2
```

[Figure 3-49](#) on page 218 shows the coverage results after executing the above command.

Figure 3-49 Coverage Report After Excluding cg.B and cg.B2

Number of covered cover bins: 6 of 44 Number of uncovered cover bins: 38 of 44			
Name	Average, Covered Grade	Line	Source Code
cginst1	43%, 14% (6/44/12)	187	covergroup cg @ (negedge clk);
--A	100% (2/2)	191	A: coverpoint TDSP_CORE_GLUE_INST.opb
cginst1.A.scalar_min	100% (358/3)	196	bins scalar_min = {0};
cginst1.A.scalar_max	100% (36/3)	197	bins scalar_max = {10};
cginst1.A2	28% (4/14)	201	A2: coverpoint TDSP_CORE_GLUE_INST.opb
--vector_low[18]	100% (42/40)	206	bins vector_low[] = {[18:20]};
--vector_low[19]	100% (45/40)	206	bins vector_low[] = {[18:20]};
--vector_low[20]	100% (42/40)	206	bins vector_low[] = {[18:20]};
--vector_med[20]	100% (42/40)	207	bins vector_med[] = {[20:30]};
--vector_med[21]	0% (36/40)	207	bins vector_med[] = {[20:30]};
--vector_med[22]	0% (36/40)	207	bins vector_med[] = {[20:30]};
--vector_med[23]	0% (15/40)	207	bins vector_med[] = {[20:30]};
--vector_med[24]	0% (0/40)	207	bins vector_med[] = {[20:30]};
--vector_med[25]	0% (6/40)	207	bins vector_med[] = {[20:30]};
--vector_med[26]	0% (0/40)	207	bins vector_med[] = {[20:30]};
--vector_med[27]	0% (0/40)	207	bins vector_med[] = {[20:30]};
--vector_med[28]	0% (0/40)	207	bins vector_med[] = {[20:30]};
--vector_med[29]	0% (0/40)	207	bins vector_med[] = {[20:30]};
--vector_med[30]	0% (0/40)	207	bins vector_med[] = {[20:30]};
cginst1.AXA2	0% (0/28)	211	AXA2: cross A, A2;
--scalar_min.vector_low[18]	0% (0/1)	211	AXA2: cross A, A2;
--scalar_min.vector_low[19]	0% (0/1)	211	AXA2: cross A, A2;
--scalar_min.vector_low[20]	0% (0/1)	211	AXA2: cross A, A2;
--scalar_min.vector_med[20]	0% (0/1)	211	AXA2: cross A, A2;

Coverpoints B and B2 have been excluded from the report. When you exclude a coverpoint, the bins within that coverpoint automatically get excluded.

To list the excluded covergroup items, use:

```
report -detail -excludes -metrics covergroup -inst dtmf_recvr_core.TDSP_CORE_INST
```

[Figure 3-50 on page 218](#) shows the excluded covergroup items.

Figure 3-50 Excluded Covergroup Items

Number of excluded cover bins: 12			
Name	Average, Covered Grade	Line	Source Code
cginst1	43%, 14% (6/44/12)	187	covergroup cg @ (negedge clk);
--B	EXCL	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
cginst1.B.auto[0:429496728]	P-EXCL	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
cginst1.B.auto[429496729:858993458]	P-EXCL	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
cginst1.B.auto[858993458:1288490186]	P-EXCL	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
cginst1.B.auto[1288490187:1717986915]	P-EXCL	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
cginst1.B.auto[1717986916:2147483644]	P-EXCL	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
cginst1.B.auto[2147483645:2576980373]	P-EXCL	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
cginst1.B.auto[2576980374:3006477102]	P-EXCL	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
cginst1.B.auto[3006477103:3435973831]	P-EXCL	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
cginst1.B.auto[3435973832:3865470560]	P-EXCL	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
cginst1.B.auto[3865470561:4294967295]	P-EXCL	212	B: coverpoint TDSP_CORE_GLUE_INST.opb
cginst1.B2	EXCL	216	B2: coverpoint TDSP_CORE_GLUE_INST.opb
--opbmin	P-EXCL	220	bins opbmin = {0};
--opbmax	P-EXCL	221	bins opbmax = {60};

In the above report:

- Coverpoints B and B2 show as EXCL (to indicate excluded).
- Bins within the coverpoints show as P-EXCL (to indicate that these items are excluded because the parent was excluded).



Items marked P-EXCL are excluded because the parent was excluded. You cannot *un-exclude* these items individually. To *un-exclude* these items, you must *un-exclude* the parent.

Similarly, you can exclude more coverpoints.

3.2.8.4 Excluding Coverbins

To exclude specific coverpoint bins from coverage analysis, use the `-coverbin` option of the `exclude` command, as shown below:

```
exclude [-type <type> | -inst <instance>]
-coverbin <covergroup_name>.<coverpoint_name>.<coverbin_name>
```

For example, to exclude coverbin `vector_low[20]` of coverpoint A2, use:

```
exclude -inst dtmf_recv_core.TDSP_CORE_INST -coverbin cg.A2.vector_low\[20\]
```

Or

```
exclude -inst dtmf_recv_core.TDSP_CORE_INST -coverbin
{"cg"."A2"."vector_low[20]"}
```

[Figure 3-51](#) on page 219 shows the coverage report after executing the above command.

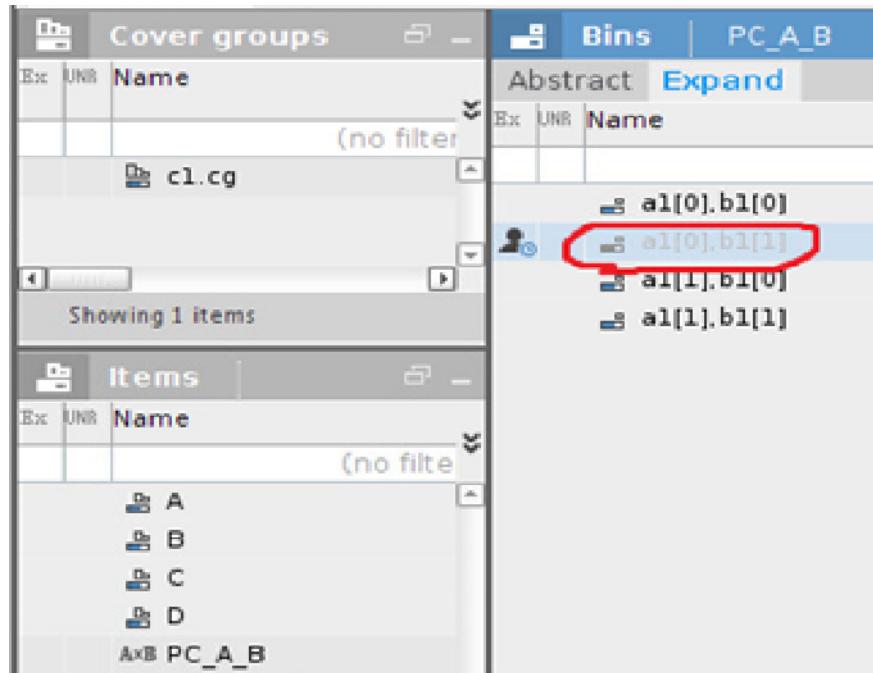
Figure 3-51 Coverage Report After Excluding vector_low[20]

Name	Average, Covered Grade	Line	Source Code
cginst1	41%, 12% (5/43/13)	187	covergroup cg @ (negedge clk);
-A	100% (2/2)	191	A: coverpoint TDSP_CORE_GLUE_INST.opa
cginst1.A.scalar_min	100% (358/3)	196	bins scalar_min = {0};
cginst1.A.scalar_max	100% (36/3)	197	bins scalar_max = {10};
cginst1.A2	23% (3/13/1)	201	A2: coverpoint TDSP_CORE_GLUE_INST.opa
-vector_low[18]	100% (42/40)	206	bins vector_low[] = {[18:20]};
-vector_low[19]	100% (45/40)	206	bins vector_low[] = {[18:20]};
-vector_low[20]	100% (42/40)	207	bins vector_low[] = {[20:30]};
-vector_low[21]	0% (36/40)	207	bins vector_low[] = {[20:30]};
-vector_low[22]	0% (36/40)	207	bins vector_low[] = {[20:30]};

Coverbin `vector_low[20]` is no longer listed in the coverage report.

The syntax for excluding a cross bin is same as excluding a cover bin. For example, consider the cross bin shown in [Figure 3-52](#) on page 220.

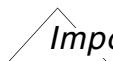
Figure 3-52 Cross bin exclusion



To exclude cross bin `a1[0].b1[1]`, use:

```
exclude -inst top -coverbin 'c1.cg'.PC_A_B.'a1\[0\],b1\[1\]'
```

Similarly, you can exclude more bins, as required.



Important

The `exclude` command allows you to exclude only the bins that are shown in detailed coverage report. The uncovered automatically generated cross bins, by default are not printed in coverage reports and therefore, such bins cannot be excluded. You can enable printing of uncovered automatically generated cross bins using the `cross_num_print_missing` covergroup option, or by using the `set_covergroup -print_all_cross_tuples` command at elaboration. You can exclude these uncovered automatically generated cross bins after they appear in the report. For more details on covergroup options and the `set_covergroup` command, see the *ICC User Guide*.

3.2.8.5 Excluding All Assertions and Covergroups

To exclude all assertions and covergroups of a specific instance or type, use the `exclude` command as:

```
exclude [-type <type> | -inst <instance>] -metrics functional
```

For example, to exclude all assertions and covergroups of instance `dtrmf_recv_core.ARST_INST`, use:

```
exclude -inst dtrmf_recv_core.ARST_INST -metrics functional
```

3.2.9 Excluding Unreachable Items

In UNR flow of IEV, if certain blocks, expressions, signals, or assertions are determined to be unreachable, then those items are marked UNR by IEV. Using IMC, you can exclude those UNR items.

To exclude the UNR items, use the `-unr` option with the `exclude` command.

Consider a sample Block coverage report shown in [Figure 3-53](#) on page 221.

Figure 3-53 Block Coverage Report

Instance name: top.c1					
Type name: can_dataer					
File name: /net/hdlsop5/export/home/ruchikas/testcases_IMC/unr_flow_example					
Number of covered blocks: 4 of 5					
Number of uncovered blocks: 1 of 5					
Number of excluded blocks: 0					
Number of unreachable blocks: 1					
Count	Block	Line	Kind	Origin	Source Code
1	1	9	code block	8	if (reset)
1	2	10	true part of	9	left <= 0;
1	3	11	false part of	9	else if (load && dispense)
UNR	4	12	true part of	11	left <= data;
1	5	11	implicit else	11	else if (load && dispense)

To exclude the unreachable block of instance `top.c1` from coverage results, use:

```
exclude -inst top.c1 -block 4 -unr
```

[Figure 3-54](#) on page 222 shows the report after executing the above command.

Figure 3-54 Report After Excluding Block 4

Instance name: top.cl
Type name: can_dataer
File name: /servers/scratch03/ruchikas/ruchikas/testcases_IMC/unr_flow_example/unr_f1
Number of covered blocks: 4 of 4
Number of uncovered blocks: 0 of 4
Number of excluded blocks: 1
Number of unreachable blocks: 0
Count Block Line Kind Origin Source Code
1 1 9 code block 8 if (reset)
1 2 10 true part of 9 left <= 0;
1 3 11 false part of 9 else if (load && dispense)
U-EXCL 4 12 true part of 11 left <= data;
1 5 11 implicit else 11 else if (load && dispense)

UNR block marked U-EXCL

Block 4 is now marked as U-EXCL.

Note: The UNR markers are also shown in the HTML-based report.

Consider a sample Toggle coverage report shown in [Figure 3-55](#) on page 222.

Figure 3-55 Toggle Coverage Report

Instance name: top.cl
Type name: can_dataer
File name: /servers/scratch03/ruchikas/ruchikas/testcas
Number of covered toggles: 1 of 8
Number of uncovered toggles: 7 of 8
Number of excluded toggles: 0
Number of unreachable toggles: 4
Number of signal bits partially toggled(rise): 0 of 8
Number of signal bits partially toggled(fall): 1 of 8
Hit(Full) Hit(Rise) Hit(Fall) Signal
1 1 1 clk
0 0 1 reset
UNR UNR UNR load
UNR UNR UNR dispense
0 0 0 data[1]
0 0 0 data[0]
UNR UNR 0 left[1]
UNR UNR 0 left[0]

Signals marked unreachable by IEV

To exclude unreachable signal `load`, use:

```
exclude -inst top.cl -toggle load -unr
```

To exclude only the rise transition of signal bit `left[1]`, use:

```
exclude -inst top.cl -toggle left\[1.rise\] -unr
```

To exclude all unreachable toggle transitions under instance `top.cl`, use:

```
exclude -inst top.c1 -toggle * -unr
```

The above command will exclude toggle variables `load` and `dispense` and will also exclude rise transitions of signals `left[1]` and `left[0]`.

Note: While marking UNR items as exclude, remember that a toggle bit is excluded only if both `rise` and `fall` transitions are marked as unreachable. If one of its transition (either `rise` or `fall`) is marked as unreachable, then only the unreachable transition will be marked as excluded.

For details on excluding toggle bits, see [Excluding All Signals](#) on page 198 and [Excluding Specific Signals](#) on page 199.

Note: Marking of unreachable items is not supported for enumerated toggles.

More Examples

To exclude all the UNR items of instance `top.c1`, use:

```
exclude -inst top.c1 -unr
```

The above command excludes all the unreachable blocks and expressions of instance `top.c1` from coverage results.

To exclude all of the unreachable blocks of instance `top.c1` from coverage results, use:

```
exclude -inst top.c1 -metrics block -unr
```

To exclude all of the unreachable expressions of instance `top.c1` from coverage results, use:

```
exclude -inst top.c1 -metrics expression -unr
```

To exclude UNR expression `1.1.2` of instance `top.c1` from coverage results, use:

```
exclude -inst top.c1 -expression 1.1.2 -unr
```

To exclude UNR assertion `ASSERT_PASS_TRACE_FAIL` of instance `top.M.L` from coverage results, use:

```
exclude -inst top.M.L -assertion ASSERT_PASS_TRACE_FAIL -unr
```

3.2.10 Un-Excluding Excluded Items

To *un-exclude* the excluded items, use:

```
unexclude [-type <type> | -inst <instance>]  
[-metrics <metrics_type>] | <coverage_object_specification>
```

where

```
<metrics_type> ::= [all] [block] [expression] [toggle] [fsm] [assertion]
[covergroup] [state] [transition] [arc] [code] [functional]
<coverage_object_specification> ::= -assertion <assertion_name> |
-covergroup <covergroup_name> |
-coveritem <covergroup_name>.<coverpoint_name> |
-coverbin <covergroup_name>.<coverpoint_name>.<coverbin_name> |
-fsm <fsm_name> |
-state <fsm_name>.<state_name> |
-transition <fsm_name>.<from_state>/|. <to_state> |
-arc <fsm_name>.<from_state>.<to_state>.<arc_index>
-toggle <signal_full_name> [<bit_specification>] [.rise | .fall] |
-block <index_specification> |
-expression <index_specification>

<index_specification> ::= <index> | <range>
<range> ::= <index>-<index>
<bit_specification> ::= [<index>] | [<range>]
```

See [Command Syntax](#) for details on syntax description.

This command reverses the effect of a previous `exclude` command.

3.2.11 Saving Refinements

After excluding coverage items during analysis, you can save the refinements to a file and later, in another IMC run, load the saved refinements. This saves you the effort of making exclusions again in the next IMC run.

To save refinements to a refinement file, use:

```
save -refinement <file>
```

where `<file>` is the name of the file in which the refinement information will be saved. This refinement file can be reloaded later in another IMC run to quickly apply the refinements.

Using this command, you can also save refinements from multiple refinement files to a single file.

For example, assume you applied and saved refinements in three files:

`my_exclude1.vRefine`, `my_exclude2.vRefine`, and `my_exclude3.vRefine`. To apply refinements from all the three files in the next IMC run, you need to load all the three files. After loading all the files, you can consider saving the refinements to a single file so that you do not need to load all three files again in the next IMC run.

To save the refinements from all of these files, to `my_allexclude.vRefine`, use:

```
save -refinement my_allexclude.vRefine
```

With the above command, refinements from all the three files are saved to a single file `my_allexclude.vRefine`. In the next IMC run, you can load only the `my_allexclude.vRefine` file.

3.2.12 Loading Saved Refinements

To load the already saved refinements for reuse in another IMC run, use:

```
load -refinement <file>
```

where `<file>` is the name of the file that was saved earlier in the GUI mode or batch mode of IMC.

The `load -refinement` command loads the specified refinement file, and marks the entities mentioned in the refinement file as "excluded". Excluded entities are ignored while performing grade calculations.

Note: If the underlying source code has changed, then at the time of loading the refinement file in command-line mode, a warning message is generated to indicate that some of the rules might be invalid. For more details, see [Reusing Refinements File](#) on page 226.

3.2.12.1 Loading a Manually Written Exclude File

You can also create an exclude file by manually writing the `exclude` and `unexclude` commands in it. A manually created exclude file cannot be loaded using the `load -refinement` command. To apply the exclusions stated in a manually created excluded file, use the following command at the IMC prompt.

```
source <filename>
```

Consider the following example.

File `my_excl.bat` includes following `exclude` commands:

```
exclude -inst dtmf_recv_core.AR_B_INST -toggle {present_state[1-2]}  
exclude -inst dtmf_recv_core.AR_B_INST -toggle next_state
```

To apply the above exclusions on the loaded run, use the following command:

```
source my_excl.bat
```

The above command will apply the exclusion commands written in file `my_excl.bat` to the loaded run.

3.3 Reusing Refinements File

In IMC, you can save the refinements to a file, and reuse it later (even if the design changes).

In the case of unnamed entities (for example, blocks, expressions, and arcs), there might be situations when the changes in source code or difference in CCF commands across simulations causes some of the rules to be invalid.

Note: The unnamed entities are the ones that do not have a name but are assigned a unique identifier by the tool at the time of coverage data generation. This unique identifier might change if the underlying source code changes.

In such cases, at the time of loading the refinement file, IMC notifies the user that few rules might be invalid and prompts the user on how to proceed. You can choose any of the following:

- Apply all of the rules: This will apply all of the rules from the refinement file. However, in this case, there is a possibility of incorrect markings. These markings are translated into “dirty” rules.
- Apply only the valid rules: This will apply only the valid rules from the refinement file and also highlight the invalid/orphan rules so that you can review them and then clear or apply, as required. These markings are translated into “orphan” rules.
- Cancel the operation: This will cancel the loading of refinement file.

In the case of command-line mode, IMC shows a warning message to indicate that some rules might be invalid. IMC automatically applies the valid rules.

Consider the following sequence of steps:

1. Load a run.
2. Apply refinements.
3. Save the refinement file. For example, save the refinement file as `my_exclude_blk.vRefine`.

Now, consider that there are some change in the source code of types/instances that were marked excluded or there is a difference in CCF commands across runs. Perform the following sequence of steps:

1. Regenerate the coverage data.
2. Launch IMC GUI.
3. Load the new coverage data.

4. Load the already saved refinement file (my_exclude_blk.vRefine).

As the source code has changed or the CCF file has differences, IMC displays a message, as shown in [Figure 3-56](#) on page 227.

Figure 3-56 Loading Refinement File (In the case of Invalid Rules)



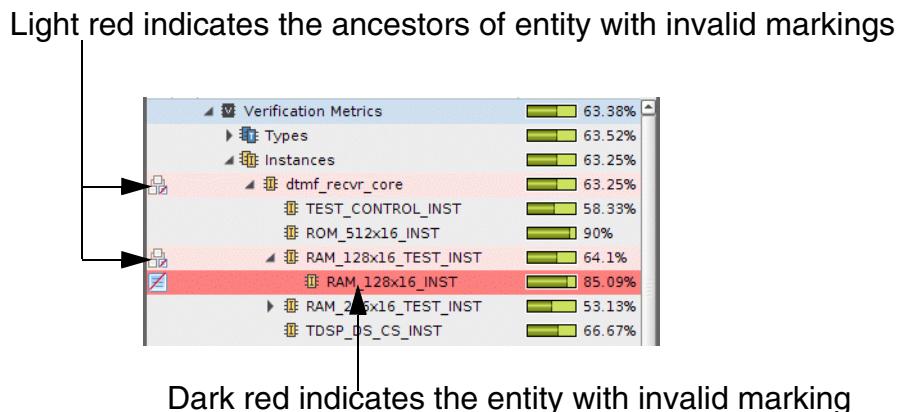
5. You can choose to:

- Apply all rules: This will apply all of the rules from the refinement file. However, in this case, there is a possibility of incorrect markings.
- Apply only valid rules: This will apply only the valid rules from the refinement file and highlight the orphan rules so that you can review them and take appropriate action.
- Cancel: This will cancel the loading of refinement file.

For example, select *Apply only valid rules*.

6. When you select *Apply only valid rules*, the entities with invalid markings are highlighted in red, as shown in [Figure 3-57](#) on page 227.

Figure 3-57 Entities with Invalid Markings

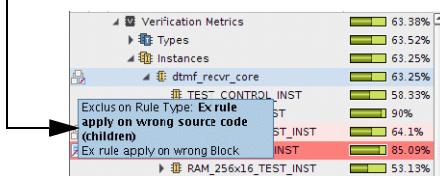


You can place the cursor on the icon shown in the *Ex* column of the entity marked red for more details.

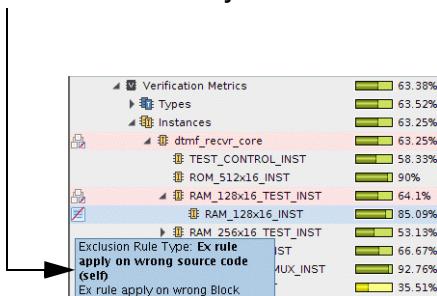
[Figure 3-58 on page 228](#) shows the tooltip shown when you place the cursor on the icon shown in the *Ex* column.

Figure 3-58 Tooltip for Invalid Markings

Tooltip shown for the parent with invalid marking



Tooltip shown for the entity with invalid marking

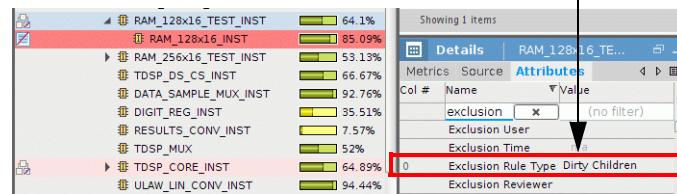


In the tooltip, text *(self)* indicates that the selected entity has invalid marking and text *(children)* indicates that some of the children might have incorrect markings.

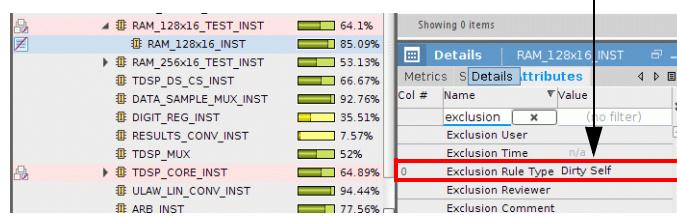
In addition, new values are added to the *Exclusion Rule Type* attribute, as shown in [Figure 3-59 on page 229](#).

Figure 3-59 Exclusion Rule Type value for Invalid Markings

Exclusion Rule Type (parent of entity with invalid marking)



Exclusion Rule Type (entity with invalid marking)



The *Exclusion Rule Type* attribute shows value — Dirty Self for the entity with invalid marking and it shows value — Dirty children for the parent of the entity with invalid marking. These values are also used at the time of applying filters. For more details on filtering, see [Filtering Values in Ex Column](#) on page 68.

Note: The dirty rules are considered orphans.

Note: At this stage, you have an option of reviewing each rule individually and then approving it or clearing it, as required (for this flow, go to [step 8](#)). You also have an option of not reviewing each rule individually and can clear or approve all the rules in the Verification Hierarchy pane together (for this flow, go to [step 7](#)).

7. You can now right-click on a instance or type highlighted as red and select any of the following actions:

- Clear all orphan rules* — To clear all the orphan rules of the selected item.
- Apply all orphan rules* — To apply all the orphan rules of the selected item and delete them. As a result, the orphan rules will be attached to the entities by index, and the rules will become applied but “dirty”.

Note: The *Apply all orphan rules* option will apply only the rules for which the index can be found. If index is not found, then no action is taken for the orphan rule.

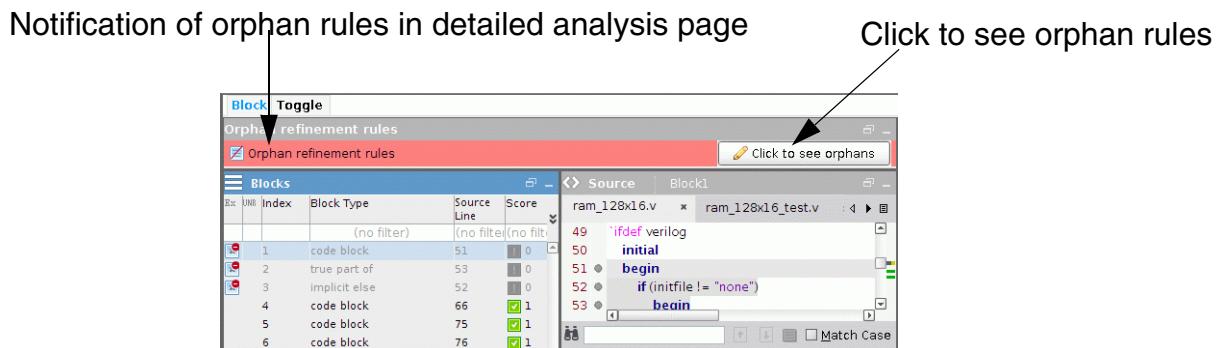
If some of the rules are marked dirty and you want to approve or clear them, then right-click the required instance or type and select any of the following:

- Clear dirty rules—To clear/delete the dirty rules of the selected instance or type.
- Approve dirty rules—To approve the dirty rules of the selected instance or type (make the rules completely valid).

8. In case you want to review each orphan rule, then open the detailed analysis page. For example, select *RAM_128x16_INST*, right-click and select *Block Analysis*.

[Figure 3-60](#) on page 230 shows the Block analysis page of instance *RAM_128x16_INST*.

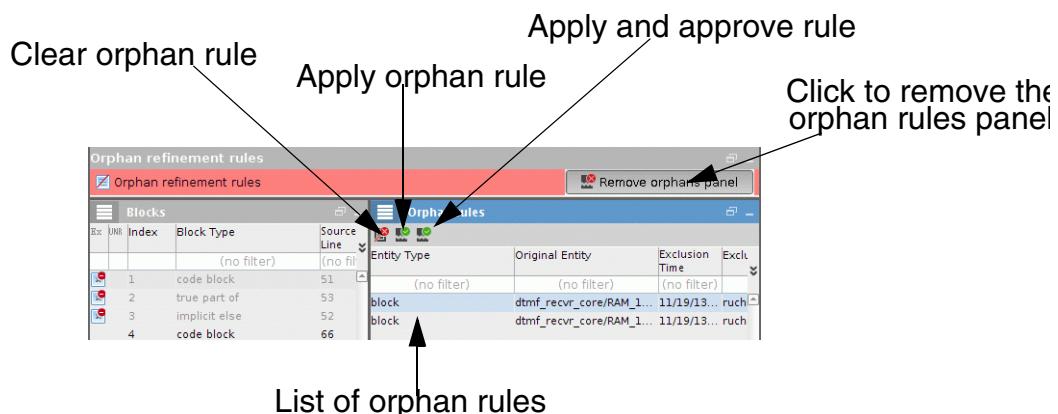
Figure 3-60 Orphan Rules Notification



9. Click the *Click to see orphans* button to review the orphan rules.

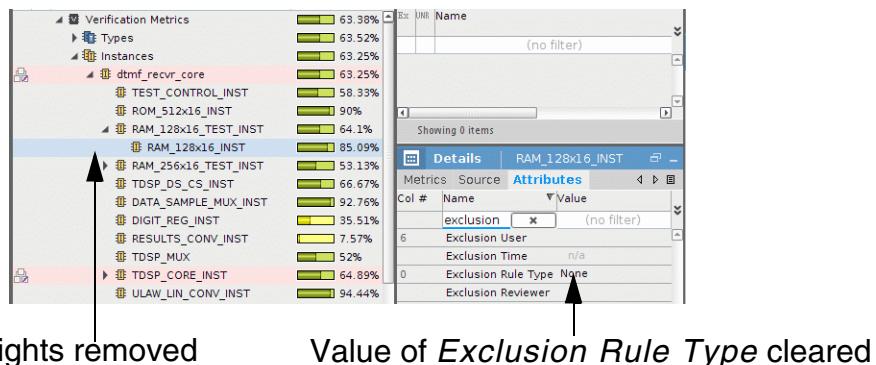
[Figure 3-61](#) on page 230 shows the details of the orphan rules.

Figure 3-61 Orphan Rules Review



10. You can review each orphan rule and perform any of the following actions from the buttons available in the table or right-click the rule and select these options from the popup menu:
 - Clear — Deletes the selected orphan rule.
 - Apply—Applies the selected orphan rule and deletes it. This option will apply only the rules for which the index can be found. If index is not found, then this option is disabled. As a result, the orphan rule will be attached to an entity by index, and the rule will become applied but “dirty”.
 - Apply and approve— Applies the selected rule and approves it (makes the rule completely valid). This option will apply and approve only the rules for which the index can be found. If index is not found, then this option is disabled.
11. When all orphan rules are cleared, the dirty flags (red highlights) on the related types/instances are automatically removed, as shown in [Figure 3-62](#) on page 231.

Figure 3-62 Orphans Cleared



You can now save the refinements file as any of the following:

- As a new refinement file: If you save the refinement file as a new refinement file, then the new refinement file will not have orphan rules and they will not be reported on loading. The original refinement file will continue to have orphan rules and if you read the original refinement file, then the orphan rules will be reported.
- Overwrite the existing refinement file: If you overwrite the existing refinement file after clearing all the orphan rules, then next time you load the refinement file, the orphan rules will not be reported.

Note: At the time of launching IMC, you can also load a TCL file (that loads and applies the exclusion rules) using the `imc -init <tcl_file>` command. The valid refinement rules are applied and the invalid/orphan rules are created and highlighted.

3.4 Converting an ICF to an IMC Exclude File

An ICCR Configuration File (ICF) cannot be used directly in IMC. This is because in ICCR, marking (refinement) was supported through `mark` and `undo_mark` commands. However, in IMC, refinement is supported through `exclude` and `unexclude` commands.

As a result, to apply marks from an ICF to a loaded run in IMC, you must convert it to an exclude file.

The `convert_icf` command of IMC allows you to quickly convert an ICF into an IMC exclude file. This command converts the `mark/undo_mark` commands written in an ICF to equivalent `exclude/unexclude` commands and saves them in a separate file.

To convert an ICF to an IMC exclude file, use:

```
convert_icf <icf_filename> -out <converted_exclude_filename> [-ies92]
[-filter <mark> | <undo_mark>]
```

where

- `<icf_filename>` is the name of the ICF that must be converted.
- `<converted_exclude_filename>` is the name of the file that will be generated after conversion.
- `-ies92` is required to make the ICF saved with IES9.2 release version compatible with the IES10.2 release version. An ICF saved with IES9.2 release version is not compatible with the IES10.2 release version due to change in expression indexing. If the ICF includes only block coverage marks, then `-ies92` option can be ignored. However, it is recommended that ICFs saved with IES9.2 release version are converted with the `-ies92` option.

For more details on ICF and marking in ICCR, see the *ICC User Guide*.

- `-filter <mark> | <undo_mark>` filters/ignores all the `mark / undo_mark` commands specified in the `<icf_filename>` during conversion.

When converting an ICF to an IMC exclude file, remember that:

- Coverage data must be loaded before executing the `convert_icf` command.
- The exclude file generated with the `convert_icf` command cannot be loaded using the `load -refinement` command of IMC. To apply the exclusions from the file generated after conversion, use the `source` command at the IMC prompt. For more details, see [Loading a Manually Written Exclude File](#).

- Currently, during conversion IMC treats both items marked as IGN and COV (using the `mark` command of ICCR) as **excluded**. For details on marking items as IGN and COV in ICCR, see the *ICC User Guide*.

Example 1 -- Apply Exclusions from an ICF Saved with IES10.2 Release Version in IMC

To apply exclusions from an ICF saved with IES10.2 release version:

1. Load the run in IMC using the `load` command.
2. Convert the ICF to an exclude file using the `convert_icf` command.
3. Load the exclude file using the `source` command.

Consider the file `mymarks.icf` that includes the following commands:

```
undo_mark -instance -b test.DUT 5
mark -ignore -instance -b test.DUT 5
undo_mark -instance -e test.DUT 2 1
mark -ignore -instance -e test.DUT 2 1
```

To convert the above ICF to an IMC exclude file, load the coverage run and then use the following command:

```
convert_icf mymarks.icf -out myexclude.bat
```

With the above command, the commands written in the `mymarks.icf` file are converted to equivalent `exclude` and `unexclude` commands and the converted commands are saved in the file `myexclude.bat`.

The following are the contents of `myexclude.bat` file.

```
unexclude -inst test.DUT -block 5
exclude -inst test.DUT -block 5
unexclude -inst test.DUT -expression 2.1.1
exclude -inst test.DUT -expression 2.1.1
```

To apply the above exclusions on the loaded run, use the following command:

```
source myexclude.bat
```

The above command will apply the exclusion commands written in file `myexclude.bat` to the loaded run.

Note: At the time of conversion, you can ignore/filter the `mark` or `undo_mark` commands. In the above example, if you want to ignore the `undo_mark` commands specified in the `mymarks.icf` file, use the following command:

```
convert_icf mymarks.icf -out myexclude.bat -filter undo_mark
```

With the above command, the `undo_mark` commands written in the `mymarks.icf` file are ignored. Only the `mark` commands are converted to equivalent `exclude` commands and the converted commands are saved in the file `myexclude.bat`.

In this case, `myexclude.bat` file will include the following commands.

```
exclude -inst test.DUT -block 5
exclude -inst test.DUT -expression 2.1.1
```

Example 2 -- Apply Exclusions from an ICF Saved with IES9.2 Release Version in IMC

To apply exclusions from an ICF saved with IES9.2 release version:

1. Load the run in IMC using the `load` command.
2. Convert the ICF to an exclude file using the `convert_icf` command.
 - a. If the ICF includes expression coverage marks, use the `-ies92` option with the `convert_icf` command.
 - b. If the ICF includes only block coverage marks, then the `-ies92` option is not required and can be ignored.
3. Load the exclude file using the `source` command.

Consider the file `mymarks.icf` that includes the following commands:

```
undo_mark -instance -e test.DUT 1 2-3
mark -ignore -instance -e test.DUT 1 2-3
undo_mark -instance -b test.DUT 7-9
mark -ignore -instance -b test.DUT 7-9
```

To convert the above ICF to an IMC exclude file, load the coverage run and then use the following command:

```
convert_icf mymarks.icf -out myexclude.bat -ies92
```

Note: The above command uses the `-ies92` option because the ICF includes expression coverage marks.

With the above command, the commands written in the `mymarks.icf` file are converted to equivalent `exclude` and `unexclude` commands and the converted commands are saved in the file `myexclude.bat`.

The following are the contents of `myexclude.bat` file.

```
unexclude -inst test.DUT -expression 1.1.2 1.1.3
exclude -inst test.DUT -expression 1.1.2 1.1.3
unexclude -inst test.DUT -block 7-9
```

Incisive Metrics Center User Guide

```
exclude -inst test.DUT -block 7-9
```

To apply the above exclusions on the loaded run, use the following command:

```
source myexclude.bat
```

The above command will apply the exclusion commands written in file `myexclude.bat` to the loaded run.

Incisive Metrics Center User Guide

IMC Command-Line Interface

This chapter discusses how to perform various tasks, such as launching IMC, loading runs, reporting, merging, and ranking metrics data through the command-line interface of IMC.

4.1 Launching IMC

IMC can be launched in any of the following mutually exclusive modes:

- GUI mode (Opens the Graphical User Interface)

To launch IMC in GUI mode, use the `imc -gui` command. This is the default mode. If `-init <tcl_file>` option is used with the `imc -gui` command, the `<tcl_file>` will be executed before launching IMC in GUI mode.

- Command-line Interactive mode (Opens the command-line interface)

To launch IMC in command-line interactive mode, use the `imc -batch` command. If `-init <tcl_file>` option is used with the `imc -batch` command, the `<tcl_file>` will be executed before launching IMC in command-line interactive mode.

- Batch mode (Allows execution of a batch script or commands and exits)

To launch IMC in batch mode, execute a batch script and exit, use the `imc -exec <command_file>` command. If `-init <tcl_file>` option is used with the `imc -exec <command_file>` command, then the `<tcl_file>` will be executed before executing the `<command_file>`.

To launch IMC in batch mode, execute the specified TCL command(s), and exit, use the `imc -execcmd <command>` command. You can specify more than one TCL command on the command line by separating them with a semicolon (;). For example, to launch IMC, run the `help` and `cdnshelp` commands and exit, use the following command:

```
imc -execcmd help;cdnshelp
```

Note: To just execute a script and exit, use the `-exec` option.



The following commands are deprecated now and an error is reported in these cases:

```
imc script.tcl  
imc -batch script.tcl
```

Instead of above commands, use the following command:

```
imc -exec script.tcl
```

4.1.1 Command Syntax

The complete syntax for the `imc` command is:

```
imc [-batch | -gui | -exec <command_file> | -execcmd <command>] [options]
```

where

- `-batch` launches IMC in Command-line Interactive mode (Opens the command-line interface).
- `-gui` launches IMC in GUI mode (Opens the Graphical User Interface).
- `-exec <command_file>` launches IMC in batch mode, executes the specified TCL file, and exits.
- `-execcmd <command>` launches IMC in batch mode, executes the specified TCL commands, and exits.

Note: If none of the options (`-batch`, `-exec`, `-execcmd`, or `-gui`) is specified, `-gui` is assumed.

- `[options]` can be any of the following:

<u><code>-help</code></u>	<u><code>-status</code></u>	<u><code>-load <run></code></u>	<u><code>-version</code></u>
<u><code>-nocopyright</code></u>	<u><code>-32bit</code></u>	<u><code>-init <file></code></u>	<u><code>-logfile <file></code></u>
<u><code>-appendlog</code></u>	<u><code>-keyfile <file></code></u>	<u><code>-appendkey</code></u>	<u><code>-verbose</code></u>
<u><code>-quiet</code></u>	<u><code>-nostdout</code></u>	<u><code>-licqueue</code></u>	<u><code>-sourcemap <file></code></u>
<u><code>-load refinement <refinement file></code></u>	<u><code>initcmd <command></code></u>	<u><code>-64bit</code></u>	<u><code>-jp</code></u>
<u><code>-display <arg></code></u>	<u><code>-memlimit <limit></code></u>	<u><code>-filter <filter></code></u>	

-help

Displays a list of the `imc` command options with a brief description about each option.

-version

Prints the IMC version number.

-status

Prints run-time statistics at the session end.

-load <run>

Loads metrics data for `<run>` while launching IMC. If the run cannot be loaded, an error is reported and the GUI is not launched.

-nocopyright

Suppresses printing of copyright banner.

-32bit

Launches the 32-bit version of the IMC executable. By default, 64-bit version of IMC is launched.

-64bit

Launches the 64-bit version of the IMC executable.

-logfile <file>

Uses `<file>` as the log file instead of the default `imc.log`.

-appendlog

Append log information from multiple runs of IMC to one log file. Use this option if you are going to run IMC multiple times, and you want all log information appended to one log file. If you do not use this option, the log file is overwritten each time you run IMC.

-keyfile <file>

Uses *<file>* as the key file instead of the default name `imc.key`.

`imc.key` is the key file that is generated automatically by IMC when you launch IMC in batch mode. This file stores all interactive commands that you have issued, including misspelled commands and the `exit` command. By default, IMC names the key file as `imc.key`. With the `-keyfile <file>` option, you can cause IMC to name the key file as *<file>* instead of the default `imc.key`.

Key files are useful when you want to reproduce an IMC session. To reproduce an interactive session, specify the name of the key file with the `-init` option. The commands in the key file are executed at the beginning of the IMC session. When IMC has processed all commands in the file, or if you interrupt processing with `CTRL-C`, input reverts back to the terminal.

You can edit the key file, as required.

-appendkey

Appends command input from multiple runs of IMC to one key file. Use this option if you are going to run IMC multiple times and you want all command input appended to one key file. If you do not use this option, the key file is overwritten each time you run IMC.

By default, IMC generates a key file called `imc.key` to capture command input.

Use the `-keyfile` option to rename the key file.

-init <file>

Executes the IMC commands in the specified *<file>* at the beginning of the IMC session.

-nostdout

Disables printing of informative summary messages to the standard output device. By default, IMC prints all messages to the standard output device as well as the `imc.log` file.

-quiet

Suppresses the printing of summary messages in the `imc.log` file. Use of this option enhances the readability of the log file because it suppresses the output of verbose informational messages.

Note: It does not suppress tool warning, error messages, tool banner, and the command-line arguments that were used to launch the tool.

-verbose

Prints informative summary messages during commands execution to the `imc.log` file.

-sourcemap <file>

Reads and applies source mappings specified in `<file>`. The `<file>` must include pair of paths, one per line. This option is useful when you want to map the source to new paths at the time of invoking IMC. To use this option, specify all the mappings in a separate file and then pass it to `imc` with the `-sourcemap` option.

For more details on mapping source paths, see [Mapping Source Path](#) on page 247.

-licqueue

Queues the request for `Affirma_sim_analysis_env` license, if not currently available, and runs the tool when the license becomes available.

-load_refinement <refinement_file>

Loads refinement file `<refinement_file>` at the time of launching IMC.

For example, to launch IMC in interactive command-line mode, and to load a refinement file named `my_exclude.vRefine` at the time of launching IMC, use:

```
imc -batch -load_refinement /home/<user_dir>/analyze/my_exclude.vRefine
```

You can load multiple refinement files using the `-load_refinement` option multiple times on the command line as shown below:

```
imc -load cov_work/mic/mic_arb1 -load_refinement fifo.vRefine -load_refinement rr_arb2.vRefine -batch
```

initcmd <command>

Executes the specified TCL command(s) at the startup of IMC.

You can specify more than one TCL command on the command line by separating them with a semicolon (;). In addition, the commands to be executed must be enclosed within double quotes (" ") or single quotes (' ').

For example, the following command launches IMC in command-line interactive mode, and then immediately loads the run `test1` and also merges the runs `test1` and `test2`.

```
imc -batch -initcmd "load -run test1; merge test1 test2 -out merged_results"
```

-jp

Enables Japanese keyboard input support and sets the default input language as Japanese.

For example, if you use the `imc -jp` command, then SCIM (Smart Common Input Method) is used as the default input engine and Japanese is set as the default language.

After this command, you can provide input to IMC in Japanese language from the keyboard.

Note: Common linux commands to set input method for all applications are:

```
setenv GTK_IM_MODULE ibus
setenv QT_IM_MODULE ibus
setenv XMODIFIERS @im=ibus
setenv LC_ALL
setenv LC_CTYPE ja_JP.utf8
```

You can set any input method (not just `ibus`) and any local type (not just Japanese).

-display <arg>

Specifies the screen on which the window must be displayed.

-memlimit <limit>

Sets the heap size of Java process. This will not change the C++ heap and will not impact coverage merging or reporting performance.

Examples:

```
imc -memlimit 4000M
imc -memlimit 4G
```

-filter <filter>

Applies changes on the main table of the view.

The format for specifying the filter is:

[attribute_name] : [operator] [value]

where operator can be any of the: {>, <, =, <=, >=, !=, range, ~, !~}

For example: imc -filter name:~comp

4.2 Loading Runs

After launching IMC, you can load a coverage run to perform various tasks. To load a coverage run, use:

```
load -run { [ [<workdir>/]<scope>/]<test>}
```

where

[[<workdir>/]<scope>/]<test> specifies the runs to be loaded. It can include just the run name (which can contain wildcard characters * and ?), or the complete path.

- <workdir> specifies the root location where all of the metric data is stored. It must contain valid filename characters and path separators. <workdir> can use both absolute paths (for example, /home/me/chip/verification/todaysrev) and relative paths (for example, verification/todaysrev).

Note: By default, when you launch IMC, the working directory <workdir> is relative to the directory in which IMC was started. You can set an alternate working directory for the subsequent commands using the `preferences -set` command. For more details, see [Setting/Viewing Preferences](#) on page 249.

- <scope> specifies a particular configuration of hardware (DUT). It must contain valid filename characters.
- <test> specifies the run name for a single simulation. It should contain valid filename characters and can contain wildcards.

Note: You can specify only one run with the `load -run` command.

Important

You cannot load more than one run in an IMC session. In the case of multiple `load -run` commands, the run specified with the last command is loaded. The runs specified with earlier `load -run` commands are automatically unloaded.

Examples

To load run TB1 from the default cov_work/scope directory, use:

```
load -run TB1
```

To load coverage from mywork/mysys1_scope/test1, use:

```
load -run ./mywork/mysys1_scope/test1
```

If the number of path characters (/) is less than two, then the missing path is assumed to be the default path, which is cov_work. For example, if <test> is specified as:

```
load -run myscope/test1
```

then the coverage data is loaded from cov_work/myscope/test1.

Note: IMC also allows loading of refinement file. For details, see [Loading and Applying Refinements](#) on page 245

4.2.1 Listing Currently Loaded Run

To view the currently loaded run, use:

```
show [-run]
```

4.2.2 Unloading Run

To unload the loaded run and associated model files, use:

```
unload [-run]
```

4.3 Merging Runs

To merge the runs:

1. Set the merge configuration using the merge_config command (optional).
2. Merge the runs using the merge command.

For details on these commands, see [Merging Runs](#) on page 259.

4.4 Ranking Runs

To rank the runs:

1. Set the ranking configuration using the `rank_config` command (optional).
2. Rank the runs using the `rank` command.

For details on these commands, see [Ranking Runs](#) on page 297.

4.5 Generating Reports

The command-line interactive mode of IMC provides following two commands for generating reports:

- `report`
- `report_metrics`

For more details on both the commands, see [Reporting and Grading](#) on page 315.

4.6 Refining Metrics Data

Refinements (exclude and un-exclude items) can be applied in IMC's GUI as well as batch mode. The `exclude` command allows you to exclude items from coverage analysis. The `unexclude` command allows you to un-exclude previously excluded items.

For details on these command, see [Refinement in Command-Line Interactive Mode](#) on page 188.

For details on applying refinements in GUI mode, see [Refinement in IMC GUI](#) on page 167.

4.6.1 Reusing Refinements

In IMC, you can save the refinements to a file, and later in another IMC run, you can load and apply the refinements saved earlier.

4.6.2 Loading and Applying Refinements

In the batch mode of IMC, you can load the refinements file that was saved earlier. To load a refinements file, use:

```
load -refinement <file>
```

where `<file>` is the name of the file that was saved earlier in the GUI mode or the batch mode of IMC.

For more details, see [Loading Saved Refinements](#) on page 225.

4.6.3 Saving Refinements to a Single File

In the batch mode of IMC, you can save refinements to a file and later, in another run of IMC, load the saved refinements. This helps in saving the effort of making exclusions again in the next IMC run.

To save refinements to a refinement file, use:

```
save -refinement <file>
```

where *<file>* is the name of the file in which the refinement information will be saved. This refinement file can be reloaded later in another IMC run to quickly apply the refinements.

For more details, see [Saving Refinements](#) on page 224.

4.7 Exporting Metrics Data to CSV File

You can use the `csv_export` command to export the metrics tree to a CSV file.

The following is the syntax of the `csv_export` command:

```
csv_export
[-bins]
[-view <view_name> [-filter <filter>]]
[-depth <tree_depth>]
[-out <output_file>]
[-overwrite]
```

where

- `-bins` exports all the cover bins in the metrics hierarchy tree to the csv file.
- `-view <view_name>` allows you to export metrics tree using a specific view that would have been created using IMC GUI. In the absence of this option, the default view *All_metrics* is used.
 - `-filter <filter>` specifies filter on the specified view. The format for specifying the filter is:

`[attribute] : [operator] [value]`

where operator can be any of the: `{>, <, =, <=, >=, !=, range, ~, !~}`

For example:

```
-filter name:~comp
```

You can also specify multiple filters on the command line. For example:

```
-filter owner:~shaim -filter status:=passed
```

Note: The filter specified on the command line will override the existing value in the view. In addition, an error is reported if the attribute_name specified with the filter does not exist.

- `-depth <tree_depth>` specifies the depth of the metrics tree to be exported. It can be any of the following:
 - level number — to indicate the number of levels to be exported to a CSV file.
 - `all`— to export all the levels (complete metrics tree) to a CSV file.
- Note:** Default value is `all`.
- `-out <output_file>` redirects the CSV file to an alternate file instead of the default `<timestamp>.csv` file.
- `-overwrite` enables overwriting of the existing output file. By default, the output file is not overwritten and an error is reported.

You can also export metrics data to a CSV file using the IMC GUI. For more details, see [Exporting Metrics Tree to a CSV File](#).

4.8 Mapping Source Path

IMC by default, looks for the design source files in the directory from which IMC is launched. If the design instrumentation location is different from the one from which IMC is launched, map the source paths so that IMC can locate the design source files. The `sourcemap` command is used to map the source to new paths, remove existing mapped sources, and list the currently mapped paths. This command is useful if files are moved, or are referenced differently on different systems.

The Backus-Naur Form (BNF) of the `sourcemap` command is:

```
sourcemap
[-add <old> <new> |
 -remove <old> <new> |
 -clear |
 -print |
 -read <file>]
```

where

- `-add <old> <new>` maps the `<old>` path to the `<new>` path.

- ❑ <*old*> indicates the path of the source files at the time of metric data generation. The <*old*> path must start at the root level.
- ❑ <*new*> indicates the new location of source files. The <*new*> path must start at the root level.

For example, to map the directory /home1 to the directory /mnt/emily/home1, use:

```
sourcemap -add /home1 /mnt/emily/home1
```

- -print lists currently mapped paths. This option is same as the command without any options.
- -clear removes all existing mapped paths.
- -remove <*old*> <*new*> allows removing of specific mapped path. For example, the following paths are mapped:

```
/home1 /mnt/sally/home1  
/home2 /mnt/sally/home2  
/home3 /mnt/emily/home3
```

To remove the path /home2 /mnt/sally/home2, use:

```
sourcemap -remove /home2 /mnt/sally/home2
```

- -read <*file*> reads and applies mapping specified in <*file*>. This option is useful when you want to apply multiple mappings using a single command. To use this option, specify all the mappings in a separate file and then pass it to the sourcemap command using the -read option.

For example, specify the following mappings in a separate file maps.txt:

```
/home1 /mnt/sally/home1  
/home2 /mnt/sally/home2  
/home3 /mnt/emily/home3
```

To apply all these mappings in an IMC run using a single command, use:

```
sourcemap -read maps.txt
```

4.9 Setting/Removing Alias for Commands

4.9.1 Setting an alias (alias)

To set an alias for a command, use:

```
alias <alias_name> <command_name> [options]
```

where,

- `<alias_name>` is the name that you want to set as an alias for the command. If only the `alias_name` is specified, then the current definition of the specified alias is printed.
- `<command_name> [options]` specifies the command for which the alias is being set. You can also alias a command along with its options.

Note: In the absence of any options to the `alias` command, a list of all the aliases is displayed.

Examples:

To set an alias, `pref` for the `preferences` command, use:

```
alias pref preferences
```

To set an alias `rd` for the command `report -summary -metrics block`, use:

```
alias rd report -summary -metrics block
```

To view a list of all aliases set, use:

```
alias
```

To view definition for a specific alias `rd`, use:

```
alias rd
```

4.9.2 Removing an alias (unalias)

To remove an alias, use:

```
unalias <alias_name>
```

where, `<alias_name>` is the name of the alias set earlier using the `alias` command.

4.10 Setting/Viewing Preferences

By default, when you launch IMC, a certain set of preferences, such as working directory, the design directory, and so on are already set to a specific value. The following table lists the preferences available in IMC and their default values.

Name	Value
<code>workDir</code>	<code>cov_work</code>
<code>scopeDir</code>	<code>scope</code>
<code>merge.ignStrongest</code>	<code>false</code>

Name	Value
reporting.itemkind	default
reporting.source	true
reporting.showempty	false
reporting.format	text
reporting.metrics	overall:code:fsm:functional
reporting.cumulative	false
reporting.local	true
reporting.append	false
reporting.cross	expand
reporting.cubeExpand	true
exclude.strict	false
exclude.suppressInfoMess	false

You can use the `preferences` command to view the list of preferences, and set a new value for specific preferences.

4.10.1 Viewing Preferences

To view a list of preferences, use:

```
preferences -get {<name>|all} -deep
```

where

- `<name>` specifies the name of the preference whose value you want to view.
- `all` returns a list of all of the preferences along with their values. If a preference has preferences nested within it, then a `{ ... }` is printed as its value to indicate that the preference has preferences nested within it. You can view the preferences nested within a preferences using the `-deep` option.
- `-deep` enables displaying of the nested preferences, if any.

Examples

To view a complete list of preferences and their values, use:

```
preferences -get all -deep
```

[Figure 4-1](#) on page 251 shows the output of the above command.

Figure 4-1 Output of preferences -get Command

```
imc> preferences -get all -deep
merge = {
  ignStrongest = false
}
reporting = {
  itemkind = "default"
  source = true
  showempty = false
  metrics = "overall:code:fsm:functional"
  cumulative = false
  local = true
  cross = "expand"
  append = false
  cubeExpand = true
}
exclude = {
  strict = false
  suppressInfoMess = false
}
workDir = "cov_work"
scopeDir = "scope"
```

To view the value of a specific preference (workDir), use:

```
preferences -get workDir
```

To view the value of a specific preference (reporting.itemkind), use:

```
preferences -get reporting.itemkind
```

4.10.2 Setting Preferences

To set a preference to a given value, use:

```
preferences -set <name> <value>
```

where

- <name> is the name of the preference whose value you want to set.
- <value> is the value you want to set for the preference.

For example, to set the preference workDir as mycovwork, use:

```
preferences -set workDir mycovwork
```

To verify the value set for the preference workDir, use:

```
preferences -get workDir
```

[Figure 4-2](#) on page 252 shows the output of the above command.

Figure 4-2 Output of preferences -get workDir Command

```
imc> preferences -get workDir
workDir = "mycovwork"
```

For subsequent IMC commands, `workDir` is considered as `mycovwork`.

4.11 Configuring Configurable Items

The `config` command allows you to view the value, set the value, restore the value of configurable items, or export non-default configuration values to a TCL file.

By default, when you launch IMC, configurable items are set to a specific value.

Using the `config` command, you can perform the following tasks:

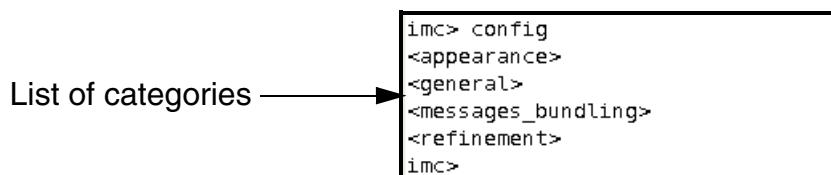
- [Viewing Configurable Items](#)
- [Setting Values for Configurable Items](#)
- [Restoring Values for Configurable Items](#)
- [Exporting Non-Default Configurable Values to a TCL File](#)

Viewing Configurable Items

The `config` command allows you to view a list of root-level categories, sub-categories, configurable items within a category, and their current values.

The `config` command without any parameters, list all the categories at the root-level, as shown in [Figure 4-3](#) on page 252.

Figure 4-3 Output of config Command



```
imc> config
<appearance>
<general>
<messages_bundling>
<refinement>
imc>
```

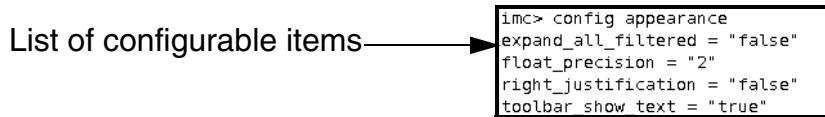
Note: The angular brackets in the command output indicate that the listed item is a category. You can view the list of items available within each of these categories by specifying the name of the category along with the `config` command.

To view a list of items within the category `appearance`, use:

```
imc> config appearance
```

[Figure 4-4](#) on page 253 shows the output of the above command.

Figure 4-4 Output of config appearance Command



The items without the angular brackets are configurable items. The values of the configurable items are listed along with the item name.

To view value of a specific item, use:

```
imc> config <config_item>
```

For example, to view the value of `source_editor`, which belongs the category `general`, use:

```
imc> config general.source_editor
```

Note: The configurable items are stated on the command-line with the complete path, which is specified as:

```
<category>.<config_item>
```

For example, to reference the configurable item `float_precision`, which is under category `appearance`, use:

```
appearance.float_precision
```

If a configurable item is within a sub-category, the path should also include the sub-category name as:

```
<category>.<sub_category>.<config_item>
```

To view a list of all the configurable items along with their current values, use:

```
imc> config -deep
```

The above command will list all the configurable items along with their current values.

Setting Values for Configurable Items

To set a value of a configurable item, use:

```
config <config_item> -set <value>
```

where

- *<config_item>* is the name (with the complete path) of the configurable item whose value you want to set.
- *-set* indicates that you want to set the value.
- *<value>* is the value you want to set for the configurable item.

For example, to set the value as 3 of configurable item `float_precision`, which belongs the category `appearance`, use:

```
imc> config appearance.float_precision -set 3
```

For example, to set the value as 0 of configurable item `float_precision`, which belongs the category `appearance`, use:

```
imc> config appearance.float_precision -set 0
```

This will set the precision as 0 and no decimals will be seen in the attributes that show values as decimals.

Note: You can also set the `float_precision` value using the *Configuration* dialog box in GUI. For more details, see [Configuring Appearance Options](#) on page 80.

Restoring Values for Configurable Items

To restore the original values of configurable item, use:

```
config [<config_item>] -restore
```

where

- *<config_item>* is the name (with the complete path) of the configurable item whose value you want to restore.
- *-restore* indicates that you want to restore the default value.

For example, to restore the original value of configurable item `float_precision`, which belongs the category `appearance`, use:

```
imc> config appearance.float_precision -restore
```

To restore the original value of all configurable items, use:

```
imc> config -restore
```

Exporting Non-Default Configurable Values to a TCL File

To export all non-default configurable values to a TCL file, use:

```
config -export <filepath>
```

where

- *-export* indicates that you want to export all the non-default configuration values to a TCL file.
- *<filepath>* is the name (with the complete path) of the file to which the configuration values will be exported.

For example, to export all non-default configurable values to a TCL file named `conf_vals`, use:

```
imc> config -export conf_vals
```

After the above command is executed, all non-default configurable values are exported to a TCL file named `conf_vals.tcl`.

Note: A `.tcl` extension is automatically added to the file name if it is not specified at the command line.

4.12 Viewing History Information

To display a list of previously executed commands in IMC, use:

```
history
```

4.13 Viewing Help

To display help for a command, use:

```
help [command]
```

This command displays description of the specified command. If no command is specified, a list of all IMC commands is displayed.

4.14 Viewing Performance Information

To display CPU performance information, use:

cpu

To display memory performance information, use:

memory

4.15 Launching Documentation

To start `cdnshelp` application and to show the specified guide, use:

```
cdnshelp
[-user_guide | -quick_ref | -kpns | -whats_new]
```

where

- `-user_guide` opens the *IMC User* guide.
- `-quick_ref` opens the *IMC Quick Reference* guide.
- `-kpns` opens the *IMC KPNS* (Know Problems and Solutions) guide.
- `-whats_new` opens the *IMC What's new* guide.

Note: If none of the options is specified with the `cdnshelp` command, by default, the *IMC User* guide is opened.

4.16 Exiting IMC

To exit from IMC, use:

exit

4.17 Using Environment Variables in CLI Commands

You can refer to environment variables as arguments in CLI commands using the standard TCL way.

For example, consider the following `load` command.

```
load -run $env(MY_RUN_DIR)
```

In the above command, `MY_RUN_DIR` is an environment variable that stores the details of path from where the run has to be loaded.

4.18 Commands Availability with Incisive_Enterprise_Simulator License

IMC requires `Affirma_sim_analysis_env` license. If the `Affirma_sim_analysis_env` license is not found, it checks for `Incisive_Enterprise_Simulator` license. If `Incisive_Enterprise_Simulator` license is found, `merge` command and a few general commands are available; however, all analysis related commands are disabled. The following commands are not available with `Incisive_Enterprise_Simulator` license:

- `save`
- `show`
- `convert_icf`
- `rank`
- `sourcemap`
- `load`
- `rank_config`
- `report`
- `exclude`
- `unexclude`
- `report_metrics`
- `unload`

Incisive Metrics Center User Guide

Merging Runs

After simulation, the metrics data is available for report generation and analysis. Coverage analysis generally is performed on aggregated data from multiple runs on one or more design configurations. Metrics data must be merged to provide a comprehensive view of what is covered.

The merge operation merges data from multiple runs from potentially different design hierarchies to a new database.

Note: The merge operation does not require loading of coverage runs.

Note: The merge functionality is available with `Incisive_Enterprise_Simulator` license. For details on commands available with `Incisive_Enterprise_Simulator` license, see [Commands Availability with Incisive Enterprise Simulator License](#) on page 257.

5.1 Merging Data

The `merge` command merges metric data of two or more runs and saves merged data as a new run.

5.1.1 Command Syntax

The complete BNF of the `merge` command is:

```
merge -out <out_path> [-overwrite]
{<runs> | -runfile <runfile>}
[-metrics <metrics_type>]
[-run_order {as_specified | by_timestamp}]
[-initial_model {empty | <runpath> | union_all | primary_run}]
[-cross_domain <icc:uxe | icc:jg | none>]
[-force_merge_versions]
[-message [<message_level>]]
[-covmodeldir <modeldir>]
```

where

```
metrics_type ::= [all] [block] [expression] [toggle] [fsm] [assertion] [covergroup]
[code] [functional]
```

```
<message_level> ::= 0 | 1
```

In the above syntax,

- `-out <out_path>` specifies where to store merged data. If the `<out_path>` argument includes just the run name, the merged output is stored in the directory set as the working directory.
- `-overwrite` enables overwriting of the existing merge output directory. If the merge output directory exists and the `-overwrite` option is used, then a warning is reported and the merge output directory is overwritten. In the absence of this option, an error is reported and the merge operation fails.
- `<runs> | -runfile <runfile>` specifies the list of runs for merging. The list of runs can be specified as any of the following:
 - At the command line using the `<runs>` option.
 - In a text file using the `-runfile <runfile>` option. The `<runfile>` is the file that includes the list of runs to be merged. The runs in `<runfile>` should be listed one per line. You can include comments in `<runfile>` using #.

Note: The primary run is determined based on the value specified with the `-run_order` option. If the `run_order` is specified as `as_specified`, then the primary run is the first run specified at the command line, or the first run in the `<runfile>`. If the `run_order` is specified as `by_timestamp`, then the primary run is determined based on the timestamp of the database model. The remaining runs that follow are the secondary runs. If * is used for `<runs>`, IMC considers the first run in the resolved list as the primary run, and remaining runs as the secondary runs.

The runs specified in `<runs>` or `<runfile>` can include just the run name or the complete path, as:

```
[ [<workdir>/]<scope>/]<run>
```

where

- `<workdir>` specifies the root location where all coverage data is stored. It should contain valid filename characters and path separators but cannot contain wildcards. `<workdir>` can use both absolute paths (for example, `/home/me/chip/verification/todaysrev`) and relative paths (for example, `verification/todaysrev`).
- `<scope>` specifies a particular configuration of hardware (DUT). It should contain valid filename characters and cannot contain path separators or wildcards.

- `<run>` specifies the run name for a single simulation. It should contain valid filename characters and can contain wildcards.
- `-metrics <metrics_type>` specifies the type of metric to merge. The `<metrics_type>` can be any, or a combination of the following:
 - all for all coverage types
 - code for block, expression, and toggle coverage
 - fsm for state, transition, and arc coverage
 - functional for both assertion and covergroup coverage
 - block for block coverage
 - expression for expression coverage
 - toggle for toggle coverage
 - assertion for PSL/SVA-based assertion coverage
 - covergroup for SystemVerilog-based covergroup coverage

If not specified, all metric types are merged.

Note: When merging specific metrics, the exclude/ignore items of only that metrics are merged. For example, if you merge block and assertion coverage, then exclude/ignore items are merged only for block and assertion coverage. The excluded and ignored items of remaining metrics will not be merged, and therefore, will not be reported.

Note: You can specify more than one metric type by separating the metric types with a colon (:). For example, to merge block and expression, use `block:expression`.

- `-run_order` specifies the order in which the runs will be processed when models are unioned. It can be specified as any of the following:
 - `as_specified`—to process the runs in the order as specified on the command line.
 - `by_timestamp`—to process the runs in reverse chronological order (that is, the run with the latest model is processed first).

Note: If not specified, the run order is assumed to be `as_specified`.

- `-initial_model` specifies the starting target model for the merge operation. The `initial_model` can be specified as any of the following:
 - `<runpath>`—Indicates that the model file of the run specified as `<runpath>` is used as the starting target model.
 - `primary_run`—Indicates that the model file of the primary run is used as the starting target model. The primary run is determined based on the selection made with the `run_order` option.

- ❑ `union_all`—Indicates that the model files of all of the runs specified on the command line with `<runs>` or in the `<runfile>` are unioned to create the starting target model.
- ❑ `empty`—Indicates that the new target model must be created based on the types and instances specified with the `-target`, `-targettype`, `-source` or `-sourcetype` options of the `merge_config` command.

Note: If not specified, the initial model is assumed to be `primary_run`.

See [Role of -resolution Union in Creation of Resultant Model](#) for details on how the final resultant model is created.

- `-cross_domain` enables standard merge between ICC runs and UXE / JasperGold runs. You can specify any of the following arguments with this option:

- ❑ `icc:uxe` to enable standard merge between ICC run and only blocks, toggles, assertions, and covergroups from UXE run.
- ❑ `icc:jg` to enable standard merge between ICC run and JasperGold run.
- ❑ `none` is the default value. It is same as not using the `-cross_domain` option.

Note: This option merges only block, toggle, assertion, and covergroup coverage from the UXE run. In addition, this option only supports standard merge. In case you want to do a union merge, then perform a union merge separately on IUS and UXE runs or IUS and JG runs, and then perform a standard run between the two unions.

Note: When using the `-cross_domain` option, it is important to provide the full IUS model with the `-initial_model <runpath>` option.

For more information on examples of this option, see [Examples: Merging Coverage Coming from ICC and UXE](#) on page 265. and [Example: Merging Coverage Coming from ICC and JasperGold \(JG\)](#) on page 266.

Note: Instead of using the `-cross_domain` option, you also have an option of enabling cross domain merge using the *Configuration* dialog box. For more details, see [Configuring Merge Options](#) on page 84.

- `-force_merge_versions` indicates that you want to merge runs generated with different major releases. By default, the merge of runs generated with different major releases is skipped. To enable merge of runs generated with different major releases, use the `-force_merge_versions` option with the `merge` command. However, such merging might produce unexpected or incorrect results.
- `-message <0 | 1>` adds diagnostic output to help locate configuration differences that limit merging. Default is 0 due to which only progress messages are output.

- `-covmodeldir <modeldir>` specifies the path where the merged `.ucm` file must be saved. By default, merge results (both `.ucm` and `.ucd` files) are stored in the output path specified with the `-out` option. If the `-covmodeldir` option is used, then the `.ucm` file is stored in the `<modeldir>`. In addition, a soft link is created into the output path (specified with the `-out` option) that will point to the `<modeldir>`.

Note: The `<modeldir>` path can be an absolute path or a relative path. If `<modeldir>` does not exist, IMC creates it, and if `imc` is unable to create it, an error is reported. In addition, if the `<modeldir>` exists but `imc` is unable to use it for saving the `.ucm` file, an error is reported.

Note: If `<modeldir>` includes just the name of the model directory, then the `merge` command creates a directory named `<modeldir>` in the current working directory.

For example, consider the following runs:

```
./cov_scope/test1/  
./cov_scope/test2/  
./cov_scope/test3/
```

To merge the coverage from above runs, use:

```
merge test1 test2 test3 -out merged_results
```

With the above command, both `.ucm` and `.ucd` files will be stored in the `./cov_scope/merged_results`.

If the `merge` command uses the `-covmodeldir` option, as shown below:

```
merge test1 test2 test3 -out merged_results -covmodeldir mod_1
```

then the `.ucm` file be stored in the `mod_1` directory. In addition, a soft link will be created from the `merged_results` directory to the `mod_1` directory.

Note: For a merge to be successful, it is important that both primary as well as secondary runs have been saved with the same set of coverage configuration file (CCF) commands. The following CCF commands, if not used in a consistent manner across runs, can prevent merging of the complete run:

- `set_code_fine_grained_merging`
- `set_merge_with_libname`
- `set_covergroup -bin_merge`
- `set_covergroup -optimize_mode`

For example, if one or more commands from the above list are used at the time of generating the primary run, then the same set of commands should be used for generating the

secondary runs that are specified on the `merge` command line. For more details on CCF commands and coverage database dumping, see the *ICC User guide*.



In case IMC is invoked from Enterprise Manager, and during the merge operation, the `-covmodeldir` option is used, then IMC ignores the path specified in the `BRUN_MODEL_DIR` environment variable and uses the path specified with the `-covmodeldir` option to save the `.ucm` file. For more details on `BRUN_MODEL_DIR` environment variable, see the *Incisive Enterprise Manager Managing Regressions* guide.

5.1.2 Merged Data Storage

The `-out` option of the `merge` command specifies where to store merged data. Consider the following runs:

```
./cov_work/scope/test1/  
./cov_work/scope/test2/  
./cov_work/scope/test3/
```

To merge coverage for the above runs to `all`, use:

```
merge test1 test2 test3 -out all
```

After the merge operation, merged results (both `.ucm` and `.ucd` files) will be stored in `./cov_work/scope/all`.

Note: As the `<out_path>` includes just the run name, the merged output is stored in the default `./cov_work/scope` directory unless the default is changed using the preferences `-set` command. For more details, see [Setting/Viewing Preferences](#) on page 249.

Examples: Merging Coverage Coming from ICC

Consider an example where `<out_path>` is specified as:

```
merge cov_work/my_design1/test1 cov_work/my_design2/test1 -out /tmp/usr/all
```

With the above command, the merged data will be stored at `/tmp/usr/all`. In this case, `tmp` is assumed to be the working directory, `usr` is considered the scope directory, and `all` is considered the merged output directory.

Consider another example where `<out_path>` is specified as:

```
merge cov_work/my_design1/test1 cov_work/my_design2/test1 -out /usr/all_merged
```

With the above command, the merged data will be stored at `/cov_work/usr/all_merged`. In this case, the working directory is not specified and therefore the default working directory `cov_work` is assumed to be the working directory, `usr` is considered the scope directory, and `all_merged` is considered the merged output directory.

Examples: Merging Coverage Coming from ICC and UXE

Following scenarios can exist:

- [Coverage from ICC and UXE are in Same test Folder](#) on page 265
- [Coverage from ICC and UXE are in Separate test Folders](#) on page 265



When merging coverage from ICC and UXE, the `-cross_domain` option is used. This option merges only block, toggle, assertion, and covergroup coverage from the UXE run. In addition, in the `merge` command, the `-initial_model <runpath>` option must be used.

Coverage from ICC and UXE are in Same test Folder

For example, coverage from ICC and UXE is stored in same test folder named `mytest`. To merge both the coverage, use the following command:

```
merge mytest -initial_model mytest -cross_domain icc:uxe -out merge_1
```

The above command merges coverage from ICC and UXE and saves the merged output in folder named `merge_1`.

Coverage from ICC and UXE are in Separate test Folders

For example, coverage from ICC and UXE is stored in separate test folders named `dbicc` and `dbuxe`. To merge both the coverage, use the following command:

```
merge dbicc dbuxe -initial_model dbicc -cross_domain icc:uxe -out merge_sep
```

The above command merges coverage from ICC and UXE and saves the merged output in folder named `merge_sep`.

Note: The primary run specified on the command line should always be the ICC run. Otherwise, a warning is reported.

Example: Merging Coverage Coming from ICC and JasperGold (JG)

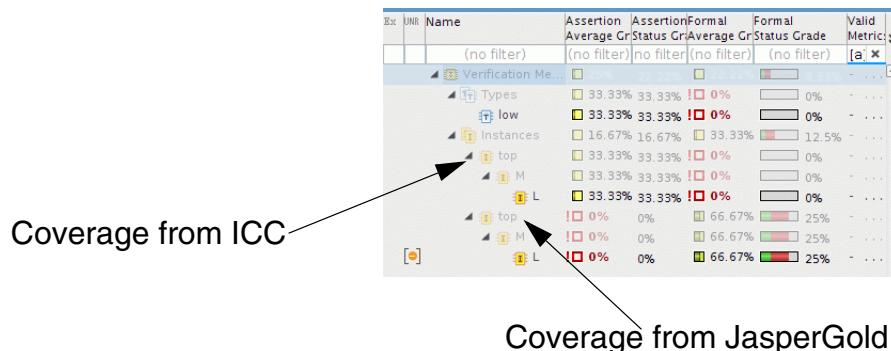
Consider that coverage from ICC is stored in `iccTest` and coverage from JG is stored in `jgTest`.

Consider the following merge command:

```
merge cov_work/scope/iccTest/ cov_work/scope/jgTest -out merged -initial_model
union_all -overwrite
```

When you load the merged database in IMC GUI, you will see the two model hierarchies side-by-side (even if they actually share similar hierarchy), as shown in [Figure 5-1](#) on page 266.

Figure 5-1 Coverage From ICC and JasperGold

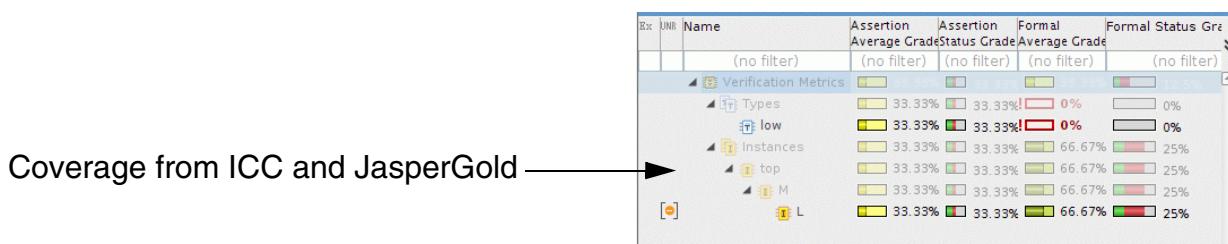


Now consider the following command that uses the `-cross_domain` merge option and performs a standard merge between ICC and JasperGold run:

```
merge iccTest jgTest -initial_model iccTest -cross_domain icc:jg -out merge_jas
-force_merge_versions -overwrite
```

The above command merges coverage from ICC and JG and saves the merged output in folder named `merge_jas`. [Figure 5-2](#) on page 266 shows the merged output in GUI.

Figure 5-2 Coverage From IUS and JasperGold (with -cross_domain)



The cross-domain merge ignores the checksums. In addition, the result of the merge is always a single coverage model having IUS domain.

5.2 Specifying Merge Configuration

IMC provides you with the `merge_config` command, using which you can:

- Specify the DUTs for Merge
- Set the Mode of Merge in Cross-Hierarchy Merging

The `merge_config` command is optional. In the absence of this command, the merge operation considers the top DUT(s) present in the primary run as the DUT to be merged.

This command is also useful when a part of the hierarchy has to be merged.

Note: You can run the `merge_config` command in the CLI mode. You can also specify the `merge_config` command(s) in a file and then specify this file in the *Mapping File* text box of the *Configuration* dialog box. For more details, see [Configuring Merge Options](#) on page 84.

5.2.1 Specify the DUTs for Merge

The `merge_config` command allows you to specify the DUT types or instances for merge. The BNF of the `merge_config` command to specify the DUTs for merge is:

```
merge_config {-source <source_instance> | -sourcetype <source_type>}  
{-target <target_instance> | -targettype <target_type>}
```

where

- `-source <source_instance>` specifies the instance that should be merged.
- `-sourcetype <source_type>` specifies the type that should be merged.
- `-target <target_instance>` specifies the instance to which the source instance/type should be merged.
- `-targettype <target_type>` specifies the type to which the source instance/type should be merged.

Note: The type specified with the `-sourcetype` and `-targettype` options should be the same. If `-sourcetype` / `-targettype` options are used, then the tool first finds all of the instances of corresponding type(s) and then, while merging each corresponding instance, it reports an error if corresponding instances being merged are not of the same type.

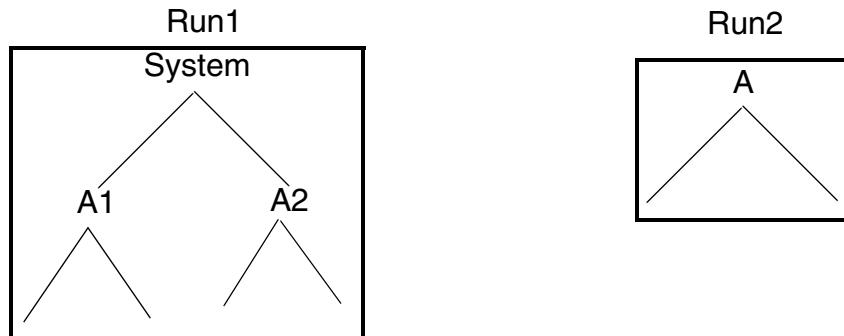
Note: The options (-source, -sourcetype, -target, and -targettype) play a vital role when the initial model is specified as empty. See [Example 3: Initial model = empty](#) on page 288 for more details on empty initial model.

Note: The syntax rules for specifying instance names and type names are same as the ones for the `exclude` command.

The `merge_config` command is additive. The final configuration is derived after considering all `merge_config` commands.

Consider the DUT models shown in [Figure 5-3](#) on page 268.

Figure 5-3 Sample DUT Models



In the above diagram, A1 and A2 are instances of module A.

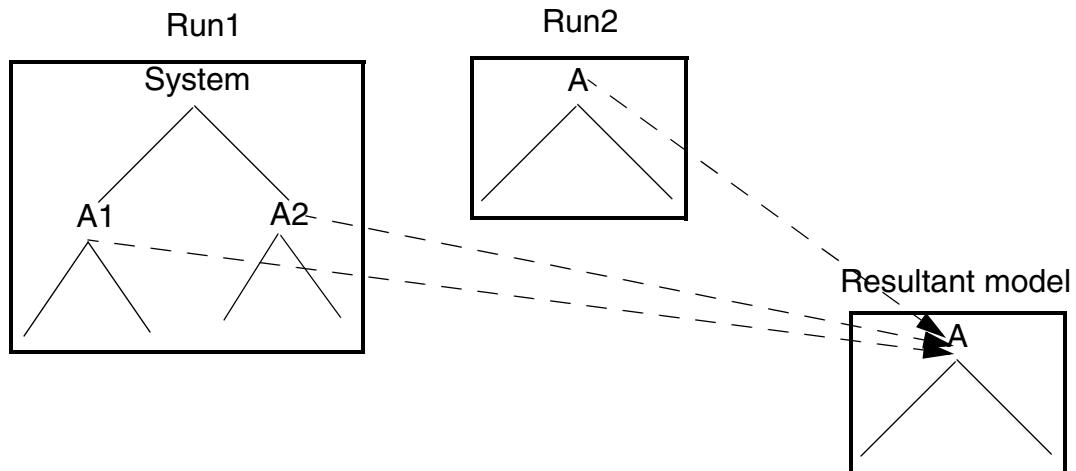
Consider the following commands:

```
merge_config -source System.A1 -targettype A
merge_config -source System.A2 -targettype A
merge Run2 Run1 -initial_model primary_run -out merged_run
```

In the above commands, primary run is specified as Run2 and the targettype is specified as A.

[Figure 5-4](#) on page 269 shows the resultant model created after executing the above commands.

Figure 5-4 Resultant Model (System-Level to Block-Level Merge)



In the above figure, notice that `System.A1` and `System.A2` are merged to `A`.

Again, consider the DUT models shown in [Figure 5-3](#) on page 268.

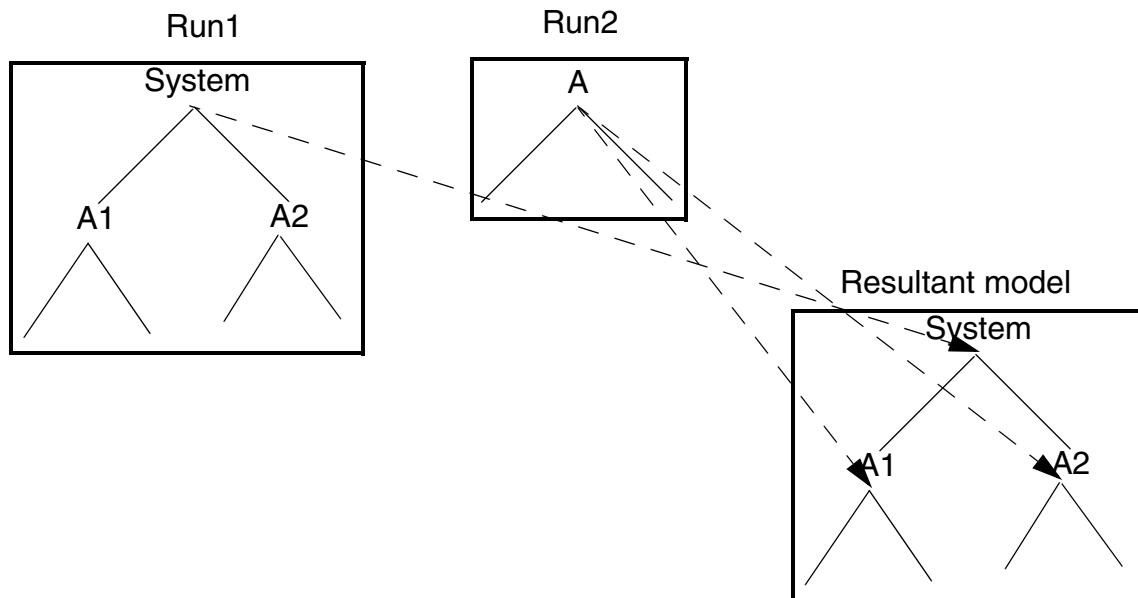
Consider the following commands:

```
merge_config -sourcetype A -targettype A
merge_Run1 Run2 -initial_model Run1 -out merged_run
```

In the above commands, primary run is specified as Run1.

[Figure 5-5](#) on page 270 shows the resultant model created after executing the above commands.

Figure 5-5 Resultant Model (Block-Level to System-Level Merge)



In the above figure, notice that `A` from the secondary run (Run2) is merged to `A1` and `A2`.

Note: Coverage items in the primary run, that are outside of the DUTs specified for merge (for example, in this case, `System`), preserve coverage from the primary run in the resultant merged coverage database.

Note: Alternatively, the following set of `merge_config` commands help achieving the same objective:

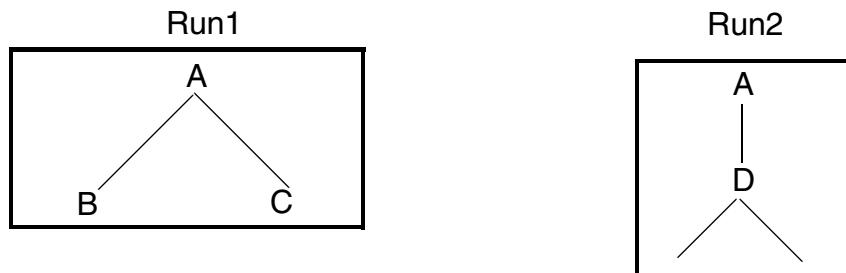
```
merge_config -sourcetype A -target System.A1
merge_config -sourcetype A -target System.A2
```

5.2.1.1 Creation of Missing Nodes

If the node specified with the `-target` option does not exist, then the missing node is created automatically in the resultant model.

Consider the DUT models shown in [Figure 5-6](#) on page 271.

Figure 5-6 Sample DUT models



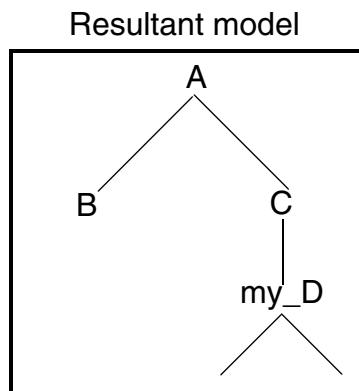
Consider the following commands:

```
merge_config -source A.D -target A.C.my_D
merge Run1 Run2 -initial_model Run1 -out merged_result -initial_model union_all
```

In the above commands, primary run is specified as Run1 and the target is specified as A.C.my_D.

[Figure 5-7](#) on page 271 shows the resultant model created after executing the above commands.

Figure 5-7 Resultant Model (Missing Node Created)



In the above resultant model, notice that the missing node A.C.my_D is created automatically.

Again, consider the DUT models shown in [Figure 5-6](#) on page 271.

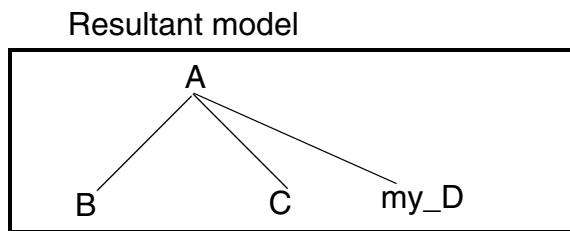
Consider the following commands:

```
merge_config -source A.D -target A.my_D
merge Run1 Run2 -initial_model Run1 -out merged_run -initial_model union_all
```

In the above commands, primary run is specified as Run1 and the target is specified as A.my_D.

[Figure 5-8 on page 272](#) shows the resultant model created after executing the above commands.

Figure 5-8 Resultant Model (Missing Node Created)



In the above resultant model, notice that the missing node A.my_D is created automatically.

Note: The missing nodes will not be created for types specified with the `-targettype` option.

5.2.2 Set the Mode of Merge in Cross-Hierarchy Merging

The `merge_config` command allows you to specify the mode of merge for the specified source/target nodes. The mode of merge for the specified source/target node can be *standard* (also known as *initial*) or *union*. The behavior of the merge operation depends on the mode set for the merge.

The BNF of the `merge_config` command to specify the mode of merge is:

```
merge_config [-resolution {initial | union}]
```

where `-resolution` specifies the mode of merge. The resolution can be set as any of the following:

- `initial` enables *standard* mode of merge for the specified target node.
- `union` enables *union* mode of merge for the specified target node.

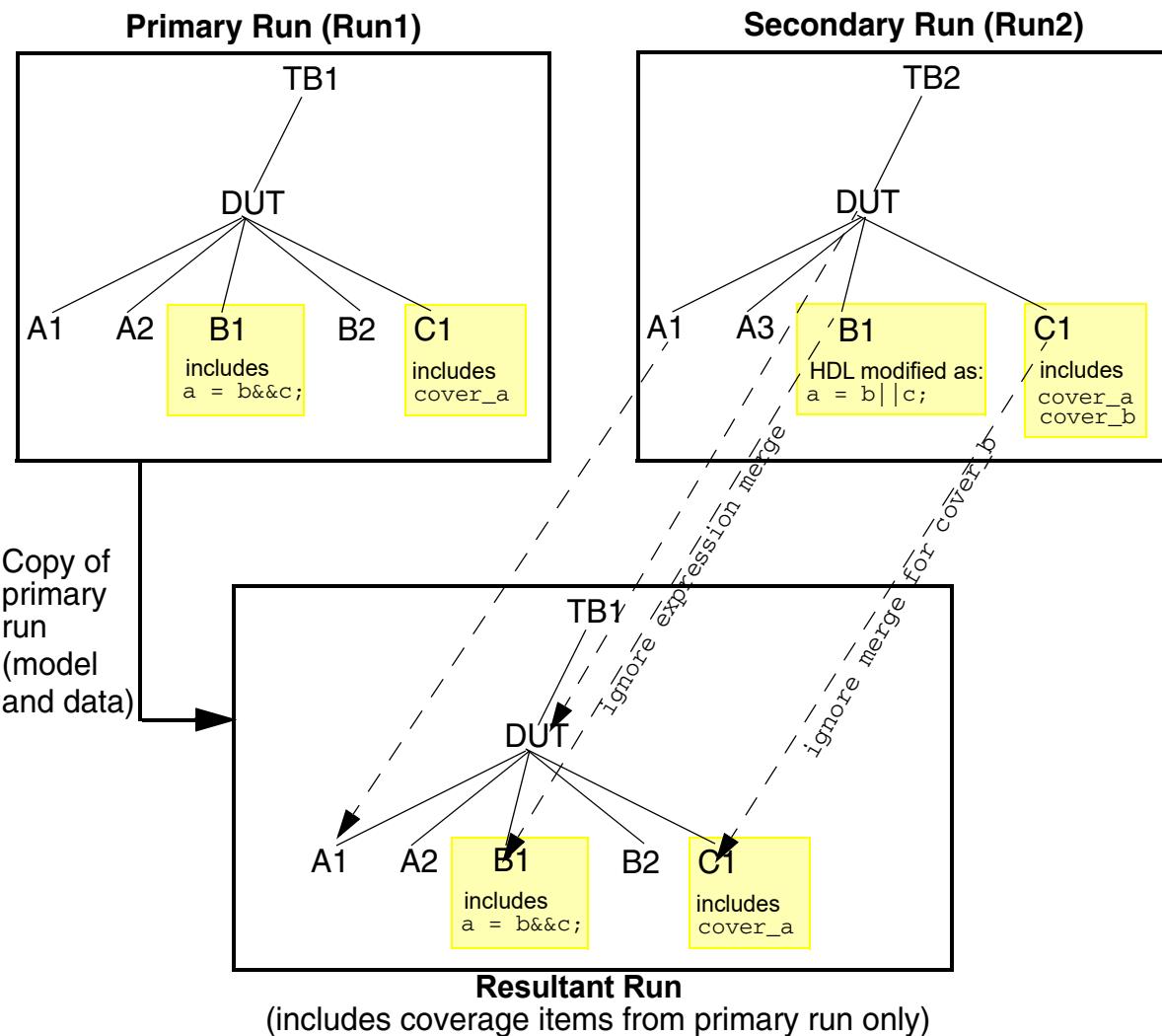
Note: If not specified, the resolution is assumed to be `initial`.

5.2.2.1 Standard Mode of Merge for Specified Hierarchy

The standard mode of merge is the default mode of merge.

Figure 5-9 on page 273 shows an example of standard mode of merge.

Figure 5-9 Merge—Standard Mode



The following `merge_config` and `merge` commands are used in this example:

```
merge_config -source TB2.DUT -target TB1.DUT -resolution initial
merge Run1 Run2 -initial_model Run1 -out result
```

In a standard merge, the merge operation creates a copy of the primary run, which is the resultant run, and then merges secondary runs in it. In this example, a copy of Run1 is created to create a resultant model. The data of Run1 is also copied to the resultant model. Data of specified DUTs from the secondary runs is then projected to the resultant model. In this example, data is projected as:

- `TB2.DUT --> TB1.DUT`

- TB2.DUT.A1 --> TB1.DUT.A1
- Since the expression in TB2.DUT.B1 is modified, all coverages are merged except for expression coverage for TB2.DUT.B1 to TB1.DUT.B1
Note: Expression coverage merge for instance B1 is ignored because HDL description differs (expression is modified in secondary run).
- TB2.DUT.C1 --> TB1.DUT.C1 (Ignoring merge of cover_b)

Note: In this mode, the merge operation merges coverage for items in the secondary runs that exist with the same definition as in the primary run. In addition, only the sampled (covered) items from the secondary coverage database are considered for merge. If a coverage item is considered for merge, then it is either merged, or a merge conflict is reported for it.

Merge Behavior if Bin-Level Merging is Enabled

Bin-level merging enables the merge operation to perform merge at the bin level instead of the default coverpoint level.

In the default merge mode, the merge operation performs a merge at the coverpoint level. In this case, the merge of the complete coverpoint is ignored if any of the bins within that coverpoint differs across runs.

In bin-level merging, the bins in the secondary runs are merged to the target/resultant model only if its name and its valid values have not been modified in the secondary run. In case there is a change in bin name or its valid values, a merge conflict is reported to indicate if there is a change in bin description or if it is not found in the resultant model. The bins that have been modified across runs are not merged. Only the bins that are same across runs are merged.

Note: Bin-level merging is supported only for SystemVerilog covergroups. In addition, bin-level merging is supported only with standard mode of merge.

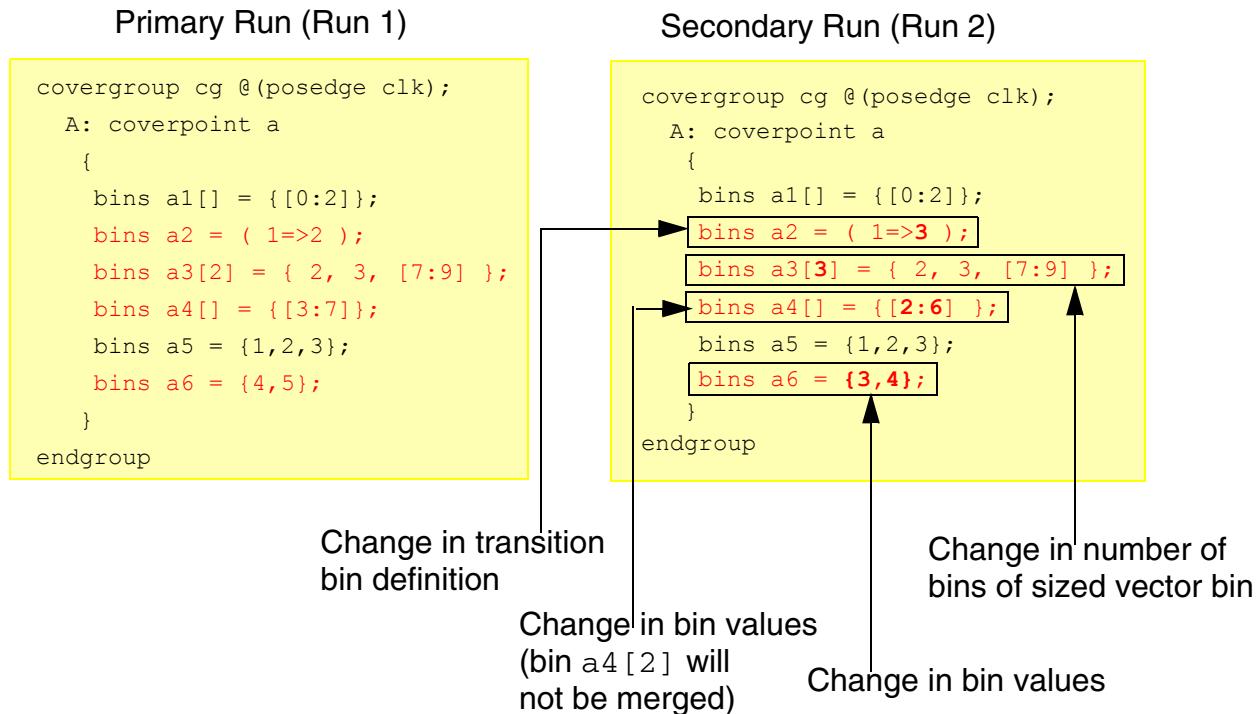


Bin-level merging is enabled using the `set_covergroup -bin_merge` command in the coverage configuration file at elaboration. For bin-level merge to be successful, it is important that both primary and secondary runs are dumped with the `set_covergroup -bin_merge` command during elaboration. For more details on this command, see the *ICC User Guide*.

Example 1: Modification of a Valid Bin

Consider the example shown in [Figure 5-10](#) on page 275.

Figure 5-10 Sample Code—Run 1 and Run 2



[Figure 5-11](#) on page 276 shows the coverage results from Run 1, Run 2, and merged coverage.

Figure 5-11 Merge—Bin-level Merging

Coverage from Run 1 (Primary)

Name	Grade
cg	92% (12/13)
--A	92% (12/13)
--a1[0]	100% (2/1)
--a1[1]	100% (1/1)
--a1[2]	100% (1/1)
--a2	100% (1/1)
--a3[2:3]	100% (2/1)
--a3[7:9]	100% (1/1)
--a4[3]	100% (1/1)
--a4[4]	100% (2/1)
--a4[5]	100% (1/1)
--a4[6]	100% (1/1)
--a4[7]	0% (0/1)
--a5	100% (3/1)
--a6	100% (3/1)

Coverage from Run 2 (Secondary)

Name	Grade
cg	93% (13/14)
--A	93% (13/14)
--a1[0]	100% (2/1)
--a1[1]	100% (1/1)
--a1[2]	100% (1/1)
--a2	100% (1/1)
--a3[2]	100% (1/1)
--a3[3,7]	100% (1/1)
--a3[8:9]	100% (1/1)
--a4[2]	100% (1/1)
--a4[3]	100% (1/1)
--a4[4]	100% (2/1)
--a4[5]	100% (1/1)
--a4[6]	100% (1/1)
--a5	100% (3/1)
--a6	100% (3/1)

Change in bin definition or values across runs

Merged coverage -- Resultant model

Name	Grade
cg	92% (12/13)
--A	92% (12/13)
--a1[0]	100% (4/1)
--a1[1]	100% (2/1)
--a1[2]	100% (2/1)
--a2	100% (1/1)
--a3[2:3]	100% (2/1)
--a3[7:9]	100% (1/1)
--a4[3]	100% (2/1)
--a4[4]	100% (4/1)
--a4[5]	100% (2/1)
--a4[6]	100% (2/1)
--a4[7]	0% (0/1)
--a5	100% (6/1)
--a6	100% (3/1)

Note: Bins marked Red are not merged.

In the above figure, following bins from secondary run are not merged to the resultant model:

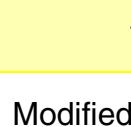
- a2—The bin values for this transition bin differs across runs.
- a3 [2]—This bin is not found in the resultant model because of the change in bin definition.
- a2 [3 , 7] —This bin is not found in the resultant model because of the change in bin definition.
- a2 [8 : 9] —This bin is not found in the resultant model because of the change in bin definition.
- a4 [2]—This bin is not found in the resultant model.
- a6—The bin values for this bin differs across runs.

Example 2: Modification of a Illegal or Ignore Bin

Consider the example shown in [Figure 5-12](#) on page 277.

Figure 5-12 Sample Code—Run 1 and Run 2

<p style="margin: 0;">Primary Run (Run 1)</p> <pre style="border: 1px solid black; padding: 10px; background-color: #ffffcc;">covergroup cg @(posedge clk); A: coverpoint a { bins a1[] = {[0:2]}; bins a2 = (1=>2); bins a3[2] = { 2, 3, [7:9] }; bins a4[] = { 4, 5 }; bins a5[] = {[3:4]}; ignore_bins e = {4,5}; } endgroup</pre>	<p style="margin: 0;">Secondary Run (Run 2)</p> <pre style="border: 1px solid black; padding: 10px; background-color: #ffffcc;">covergroup cg @(posedge clk); A: coverpoint a { bins a1[] = {[0:2]}; bins a2 = (1=>2); bins a3[2] = { 2, 3, [7:9] }; bins a4[] = { 4, 5 }; bins a5[] = {[3:4]}; ignore_bins e = {[3:4]}; } endgroup</pre>
---	---


Modified ignore bin values

In the above code, ignore bins values have changed across runs. This will result in creation of different bin spaces across runs.

[Figure 5-13](#) on page 278 shows the coverage results from Run 1, Run 2, and merged coverage.

Figure 5-13 Merge—Bin-level Merging

Coverage from Run 1 (Primary)

Name	Grade
cg	100% (7/7)
--A	100% (7/7)
--a1[0]	100% (2/1)
--a1[1]	100% (1/1)
--a1[2]	100% (3/1)
--a2	100% (1/1)
--a3[2:3]	100% (4/1)
--a3[7:9]	100% (1/1)
--a5[3]	100% (1/1)

Coverage from Run 2 (Secondary)

Name	Grade
cg	100% (7/7)
--A	100% (7/7)
--a1[0]	100% (2/1)
--a1[1]	100% (1/1)
--a1[2]	100% (3/1)
--a2	100% (1/1)
--a3[2,7]	100% (3/1)
--a3[8:9]	100% (1/1)
--a4[5]	100% (1/1)

Merged coverage -- Resultant model

Name	Grade
cg	100% (7/7)
--A	100% (7/7)
--a1[0]	100% (4/1)
--a1[1]	100% (2/1)
--a1[2]	100% (6/1)
--a2	100% (2/1)
--a3[2:3]	100% (4/1)
--a3[7:9]	100% (1/1)
--a5[3]	100% (1/1)

Note: Bins marked Red are not merged.

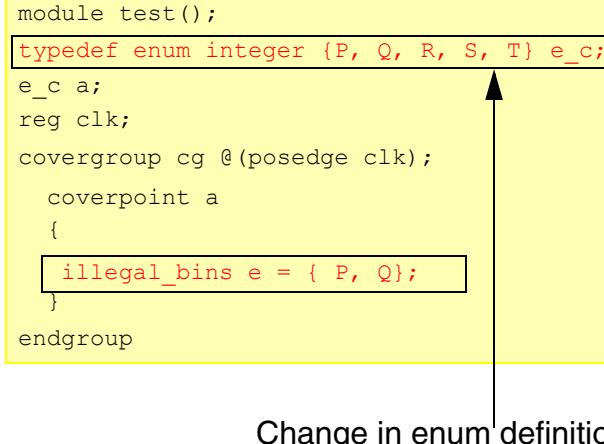
In the above figure, following bins from secondary run are not merged to the resultant model:

- a3 [2, 7] —The description of this bin differs across runs because of the difference in ignore bin values.
- a3 [8 : 9] —The description of this bin differs across runs because of the difference in ignore bin values.
- a4 [5] —This bin is not found in the resultant model because of difference in ignore bin values.

Example 3: Modification in enum Definition

Consider the example shown in [Figure 5-14](#) on page 279.

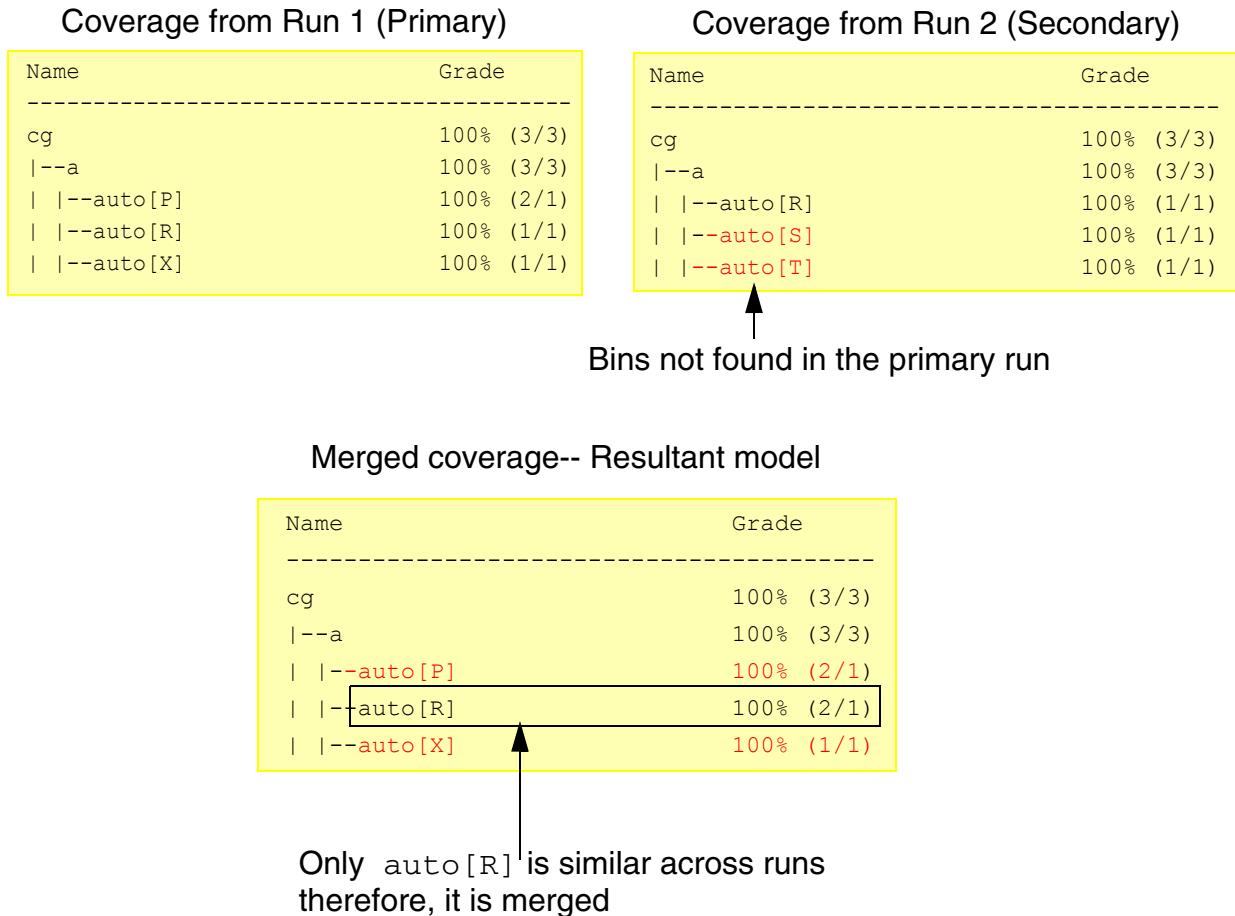
Figure 5-14 Sample Code—Run 1 and Run 2

Primary Run (Run 1)	Secondary Run (Run 2)
<pre>module test(); typedef enum integer {P, R, T, V, X} e_c; e_c a; reg clk; covergroup cg @(posedge clk); coverpoint a { ignore_bins f = { T, V }; } endgroup</pre>	<pre>module test(); typedef enum integer {P, Q, R, S, T} e_c; e_c a; reg clk; covergroup cg @(posedge clk); coverpoint a { illegal_bins e = { P, Q}; } endgroup</pre>
 <p>Change in enum definition</p>	

In the above code, the enum definition has changed across runs. In addition, the valid values of ignore or illegal bins has changed across runs. This will impact the merging of affected bins.

[Figure 5-15](#) on page 280 shows the coverage results from Run 1, Run 2, and merged coverage.

Figure 5-15 Merge—Bin-level Merging



Note: Bins marked Red are not merged.

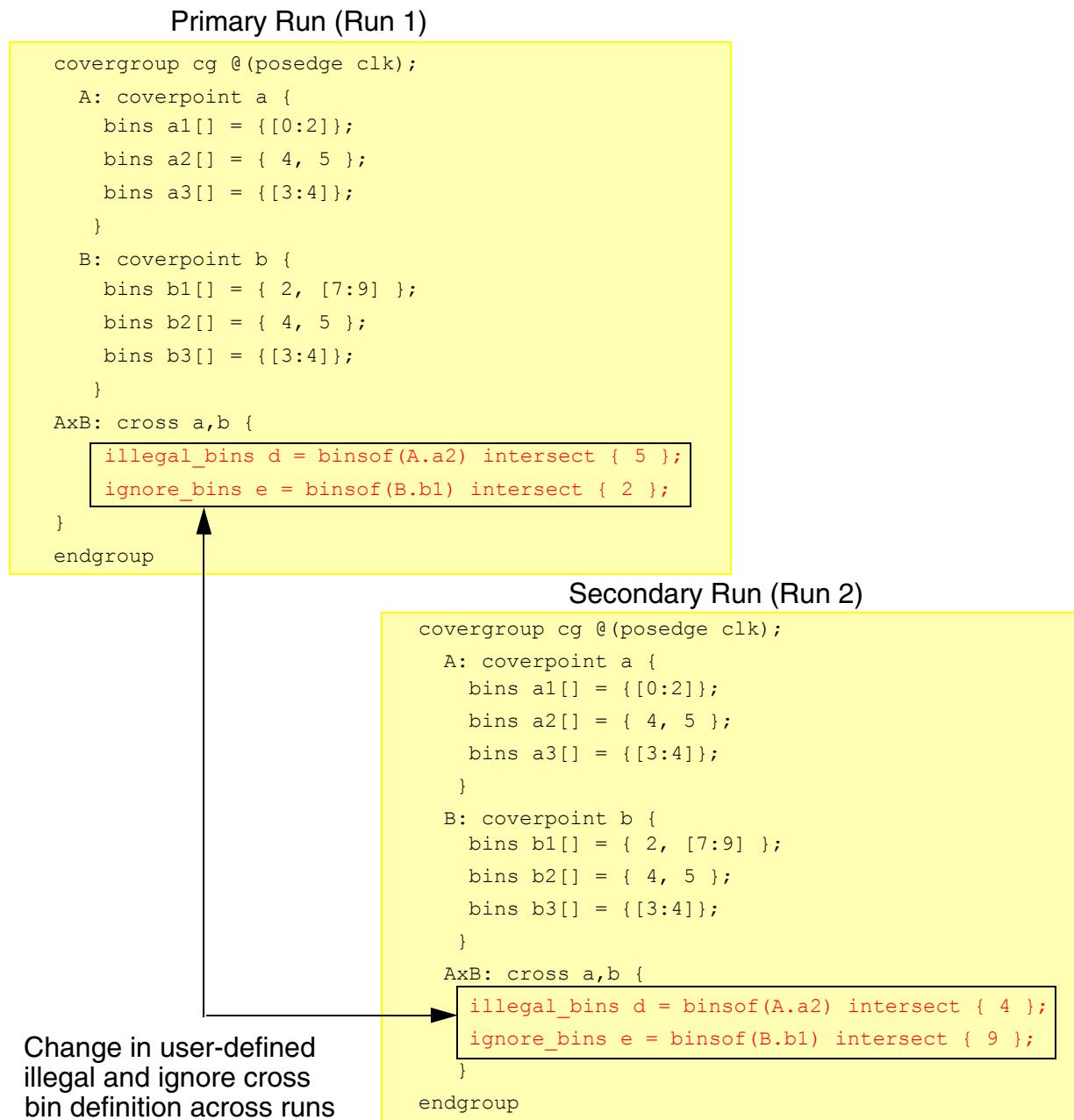
In the above figure, following bins from secondary run are not merged to the resultant model:

- auto[S] —This bin is not found in the resultant model.
- auto[T]—This bin is not found in the resultant model.

Example 4: Modification of a Illegal or Ignore User-defined Cross Bin

Consider the example shown in [Figure 5-16](#) on page 281.

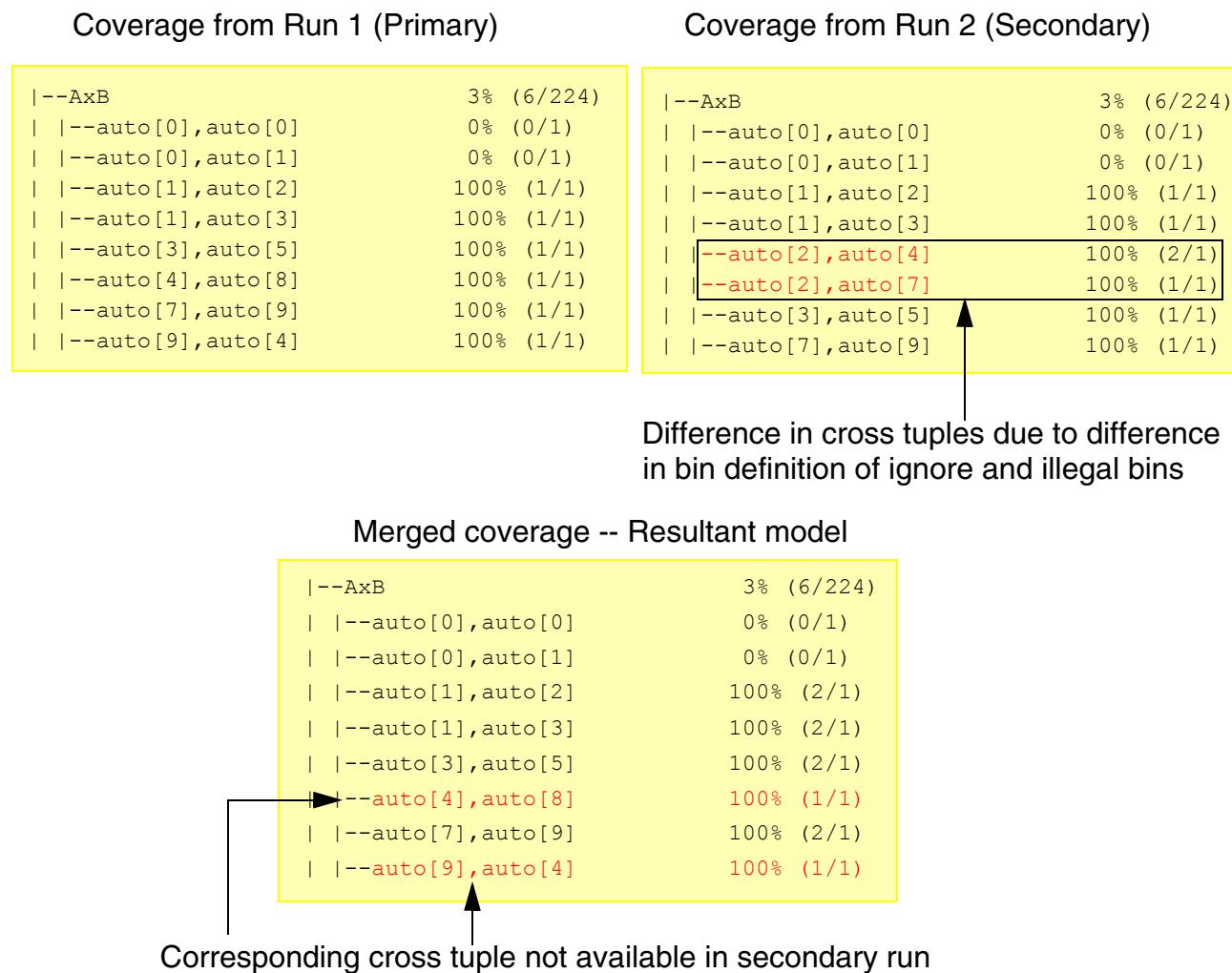
Figure 5-16 Sample Code—Run 1 and Run 2



In the above code, there is a difference in user-defined illegal and ignore cross bin definition across runs. This will result in creation of different cross tuples across runs.

[Figure 5-17](#) on page 282 shows the coverage results from Run 1, Run 2, and merged coverage.

Figure 5-17 Merge—Bin-level Merging



Note: Bins marked Red are not merged.

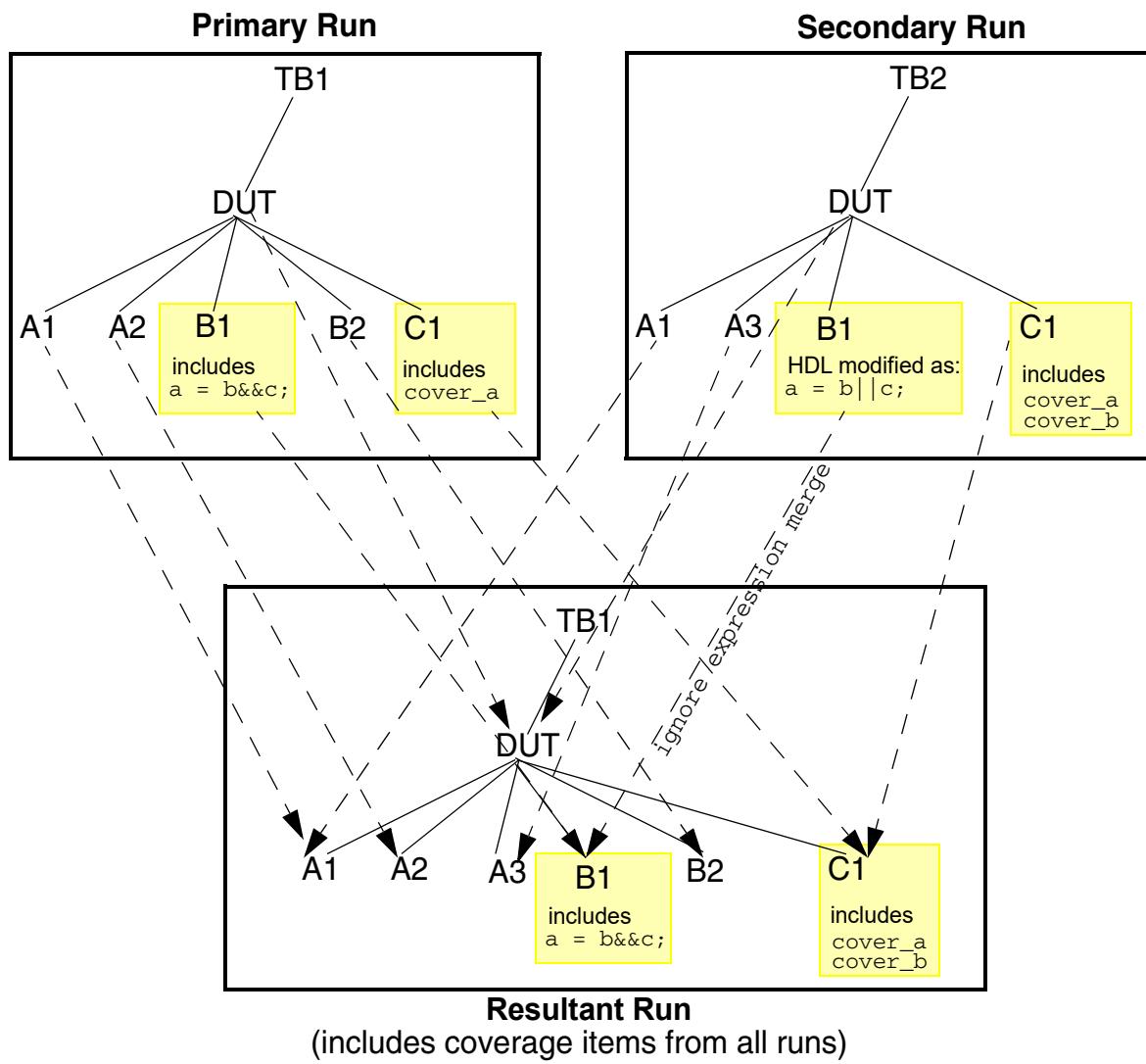
In the above figure, following bins from secondary run are not merged:

- **auto[2],auto[4]**—This bin is not found in the resultant model.
- **auto[2],auto[7]**—This bin is not found in the resultant model.

5.2.2.2 Union Mode of Merge for Specified Hierarchy

[Figure 5-18 on page 283](#) shows an example of union mode of merge.

Figure 5-18 Merge—Union Mode



The following `merge_config` and `merge` commands are used in this example:

```
merge_config -source TB2.DUT -target TB1.DUT -resolution union
merge Run1 Run2 -initial_model Run1 -out result
```

In a union merge, the models of all the specified runs are unioned to create the resultant model. In this example, the resultant model is created after unioning the models of Run1 and Run2. In this mode of merge, all coverage items from the secondary database are considered for merge. In this example, data is projected to the resultant model as:

- `TB1.DUT` and `TB2.DUT` to `TB1.DUT` of the resultant model
- `TB1.DUT.A1` and `TB2.DUT.A1` to `TB1.DUT.A1` of the resultant model

- TB1.DUT.A2 to TB1.DUT.A2 of the resultant model
- TB2.DUT.A3 to TB1.DUT.A3 of the resultant model
- TB1.DUT.B2 to TB1.DUT.B2 of the resultant model
- TB1.DUT.C1 and TB2.DUT.C1 to TB1.DUT.C1 of the resultant model
- All coverages except for expression coverage for TB1.DUT.B1 and TB2.DUT.B1 to TB1.DUT.B1 of the resultant model

Note: Expression coverage merge for instance B1 is ignored because HDL description differs (expression is modified in secondary run).

Note: The resultant model includes coverage types, non-conflicting instances, assertions, coverpoints, crosses, and covergroups that are not available in the primary model.

The union mode of merge is set by specifying the `-resolution` option of the `merge_config` command as `union`.

5.2.3 Role of `-resolution Union` in Creation of Resultant Model

The `-initial_model` option of the `merge` command specifies the starting target model for a merge operation. The final resultant model can be the same, or can be different from the starting target model depending on the value specified for the `-resolution` option of the `merge_config` command.

If the `merge_config` command is not used to set the resolution, then the final target model is same as the starting resultant model. However, if the `-resolution` option is set as `union`, then the target model is different for different scenarios.

Table 5-1 on page 284 describes the creation of resultant model in different scenarios.

Table 5-1 Creation of Resultant Model

<code>-initial_model set as:</code>	Starting Target Model	Final Resultant Model (<code>-resolution set as union</code>)
<code>primary_run</code>	Is the model of the primary run.	Is the model created after doing a union of the source and target specified in the <code>merge_config</code> command. See <u>Example 1: Initial Model = primary_run</u> on page 286 for more details.

-initial_model set as:	Starting Target Model	Final Resultant Model (-resolution set as union)
<runpath>	Is the model of the <runpath>.	Is the model created after doing a union of the source and target specified in the <code>merge_config</code> command.
union_all	Is the union of all the models.	Is the model created after doing a union of the source and target specified in the <code>merge_config</code> command. See Example 2: Initial Model = union_all on page 287 for more details.
empty	Is an empty model.	Is the model created based on the value set for the <code>-target</code> , <code>-targettype</code> , <code>-source</code> , or <code>-sourcetype</code> options of the <code>merge_config</code> command. See Scenario 1: Non-Hierarchical Name Specified with the -target Option on page 289 for more details.

Important Recommendations

- If the coverage model has not changed across runs, IMC merge is recommended to be invoked in default (`initial_model primary_run`) mode.
- If the coverage model has changed across runs and:
 - There is a single reference (or golden) coverage model containing all of the interesting coverage points for which coverage is to be merged, IMC merge is recommended to be invoked in default mode.
 - There is not a single reference (or golden) coverage database containing all of the interesting coverage points and, instead there is a need to union coverage points from different coverage databases, IMC merge is recommended to be invoked in `union_all` mode.

For example, assume that a coverage point `cp1` was present in *Day 1* coverage database but it is not present in *Day 2* coverage database. Similarly, a coverage point `cp2` is added in *Day 2* coverage database but it was not present in *Day 1* coverage database. For both of these scenarios, even though the coverage model has changed across *Day 1* and *Day 2*, if

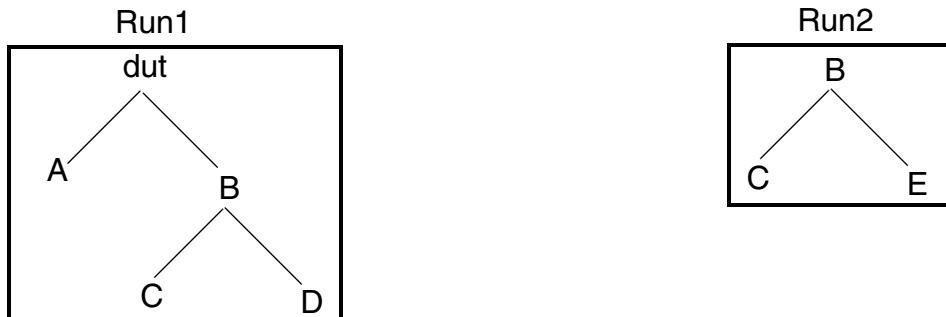
the user is not interested in coverage of `cp1` but interested in coverage of `cp2`, IMC merge is recommended to be invoked in default mode with *Day 2* coverage database being specified as the primary coverage database.

Note: The `union_all` mode is relatively compute intensive than default mode because it has an additional step to union coverage points of specified coverage databases.

5.2.3.1 Example 1: Initial Model = primary_run

Consider the DUT models shown in [Figure 5-19](#) on page 286.

Figure 5-19 Sample DUT Models

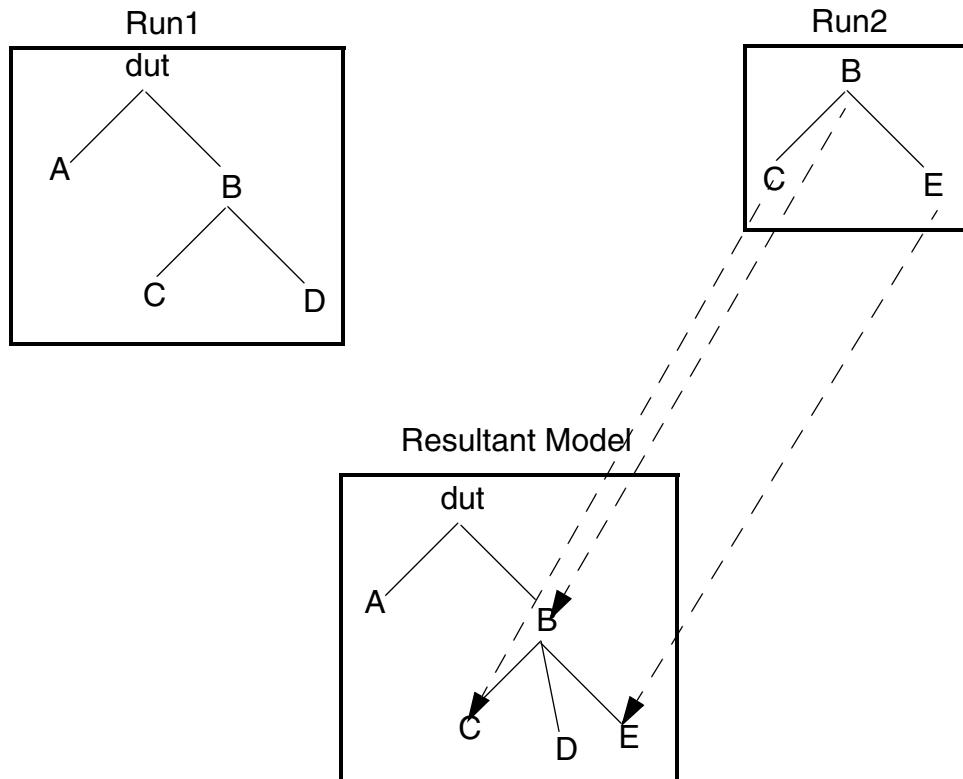


Consider the following set of commands.

```
merge_config -source B -target dut.B -resolution union
merge Run1 Run2 -initial_model primary_run
```

In the above commands, the initial model is specified as `primary_run`. Therefore, the starting target model is the model of the primary run. As the resolution is specified as `union`, the final resultant model is created after unioning the hierarchy specified through the `merge_config` command, as shown in [Figure 5-20](#) on page 287.

Figure 5-20 Resultant Model



Note: If the `-resolution` was not specified, then the resultant model would have been same as the model of the primary run.

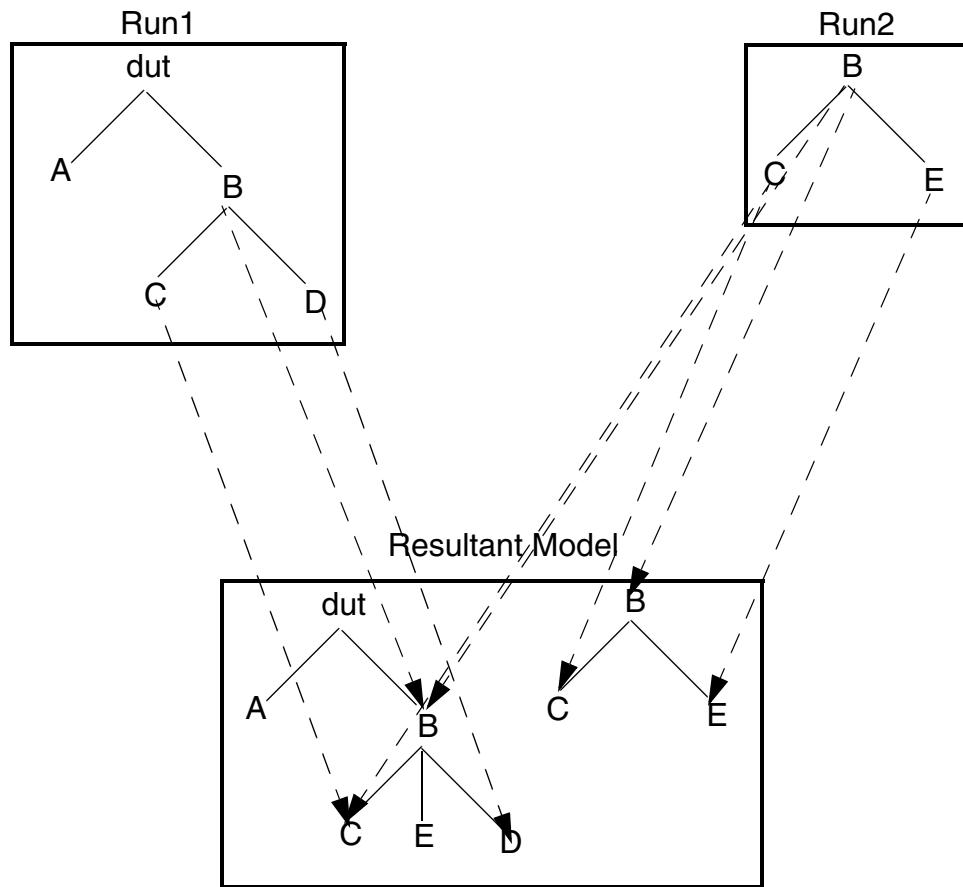
5.2.3.2 Example 2: Initial Model = `union_all`

Again consider the DUT models shown in [Figure 5-19](#) on page 286. Consider the following commands:

```
merge_config -source B -target dut.B
merge Run1 Run2 -initial_model union_all
```

In the above commands, the initial model is specified as `union_all`. Therefore, the starting target model is the model of the union of all the runs. The resultant model created in this case is shown in [Figure 5-21](#) on page 288.

Figure 5-21 Resultant Model



5.2.3.3 Example 3: Initial model = empty

If initial model is specified as `empty`, then the new target model is created based on the types and instances specified with the `-source`, `-sourcetype`, `-target`, and `-targettype` options of the `merge_config` command.

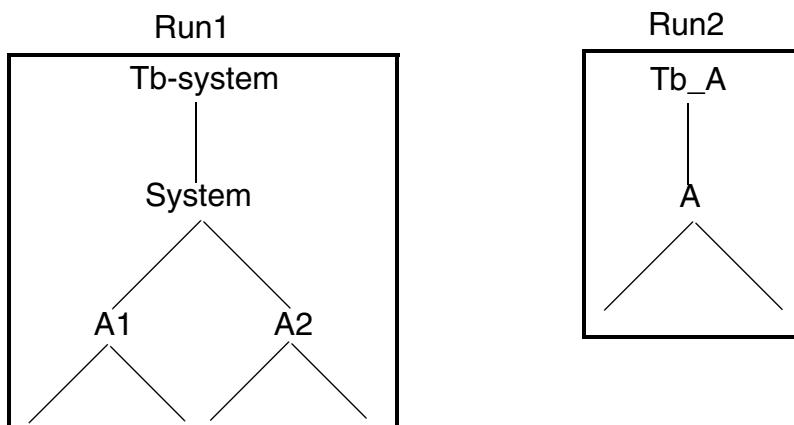
Consider the following scenarios that describe how the target model is created and how data merge happens when the initial model is `empty`:

- [Scenario 1: Non-Hierarchical Name Specified with the `-target` Option](#)
- [Scenario 2: Hierarchical Name Specified with the `-target` Option](#)
- [Scenario 3: `-targettype` Option Used](#)

Scenario 1: Non-Hierarchical Name Specified with the -target Option

If a non-hierarchical name is specified with the `-target` option, then the merge operation creates a copy of the type or instance specified with the `-source` or `-sourcetype` option at the root of the target model. The copied node is then named as the name specified with the `-target` option. Consider the runs shown in [Figure 5-22](#) on page 289.

Figure 5-22 Sample DUT Models

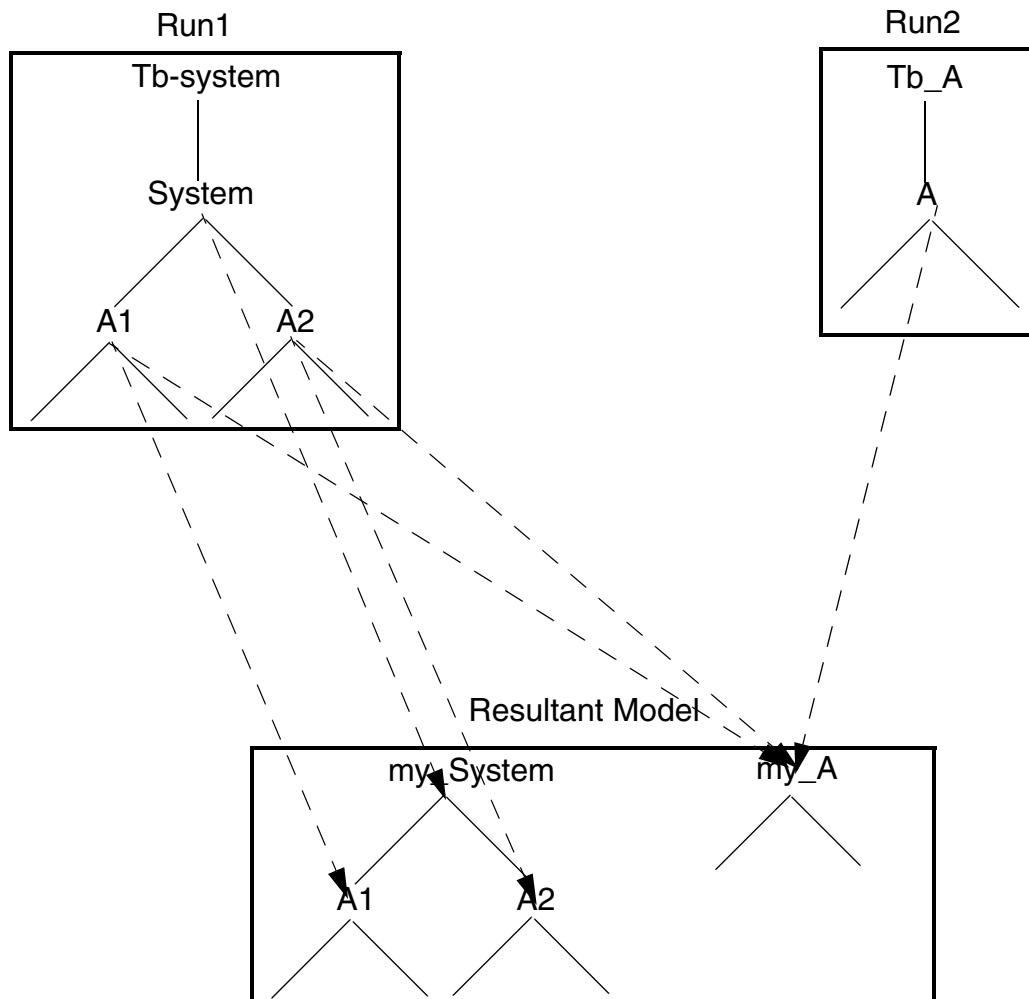


Consider the following commands to merge the runs:

```
merge_config -source Tb-system.System -target my_System
merge_config -sourcetype A -target my_A
merge Run1 Run2 -initial_model empty -out result
```

[Figure 5-23](#) on page 290 shows the resultant model created and how data merge happens with these commands.

Figure 5-23 Resultant Model (Non-Hierarchical Name with -target Option)



As `-target` is a non-hierarchical name, a copy of:

- `Tb-system.System` is created at the root of the target model and the node is named as `my_System`
- `A` is created at the root of the target model and the node is named as `my_A`

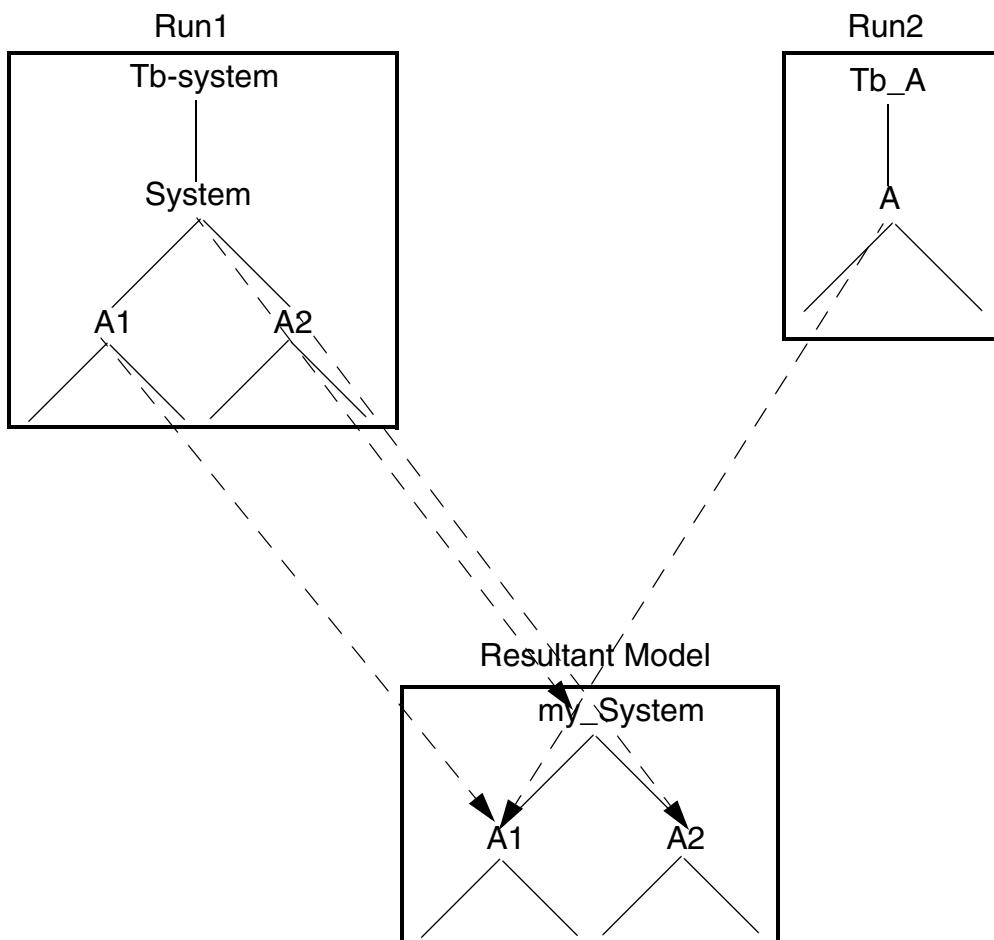
Scenario 2: Hierarchical Name Specified with the `-target` Option

If a hierarchical name is specified with the `-target` option, then the merge operation assumes that the target node already exists in the target model. If the target node exists, the source node is merged into the instance specified with the `-target` option. An error is reported if the target node does not exist in the target model. Again, consider the runs shown in [Figure 5-22](#) on page 289. Use the following commands to merge the runs:

```
merge_config -source Tb-system.System -target my_System
merge_config -source TB_A.A -target my_System.A1
merge Run1 Run2 -initial_model empty -out result
```

[Figure 5-24 on page 291](#) shows the resultant model created and how data merge happens with these commands.

Figure 5-24 Resultant Model (Hierarchical Name with -target Option)



The first `merge_config` command creates `Tb-system.System` at the root of the target model and names it as `my_System`. Notice that in the second `merge_config` command, a hierarchical name is used with the `-target` option. As a result, a new node is not created. The merge operation will assume that `my_System.A1` already exists in the target model. In this case, `my_System.A1` exists because of the use of the previous `merge_config` command. If the first `merge_config` command was not used, an error would have been reported to indicate that `my_System.A1` does not exist in the target model. The merge operation will merge `TB_A.A` to `my_System.A1` that exists in the target model.

Scenario 3: -targettype *Option Used*

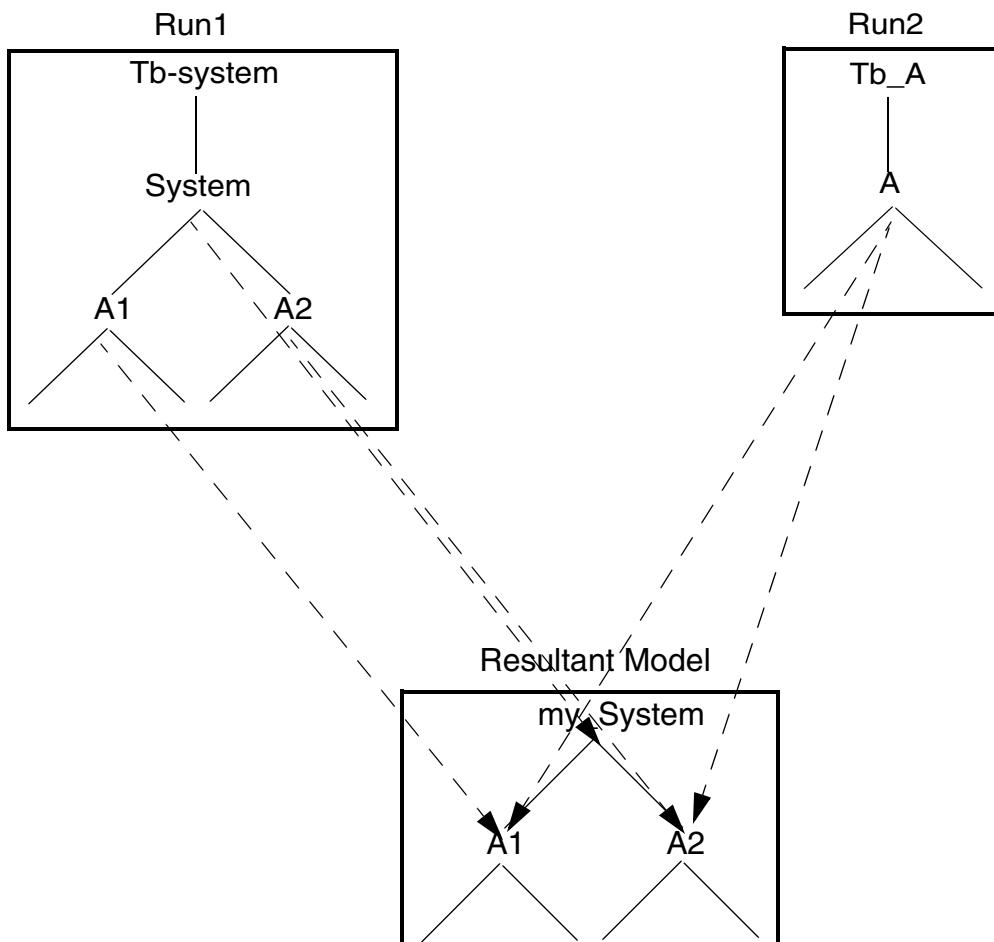
If the `-targettype` option is used, then the merge operation assumes that the target node already exists in the target model. If the target node exists, the source node is merged into each instance of the type specified with the `-targettype` option. An error is reported if the target node does not exist in the target model. Again, consider the runs shown in [Figure 5-22](#) on page 289.

Consider the following commands to merge the runs:

```
merge_config -source Tb-system.System -target my_System
merge_config -sourcetype TB_A.A -targettype A
merge Run1 Run2 -initial_model empty -out result
```

[Figure 5-25](#) on page 292 shows the resultant model created and how data merge happens with these commands.

Figure 5-25 Resultant Model (-targettype Option Used)



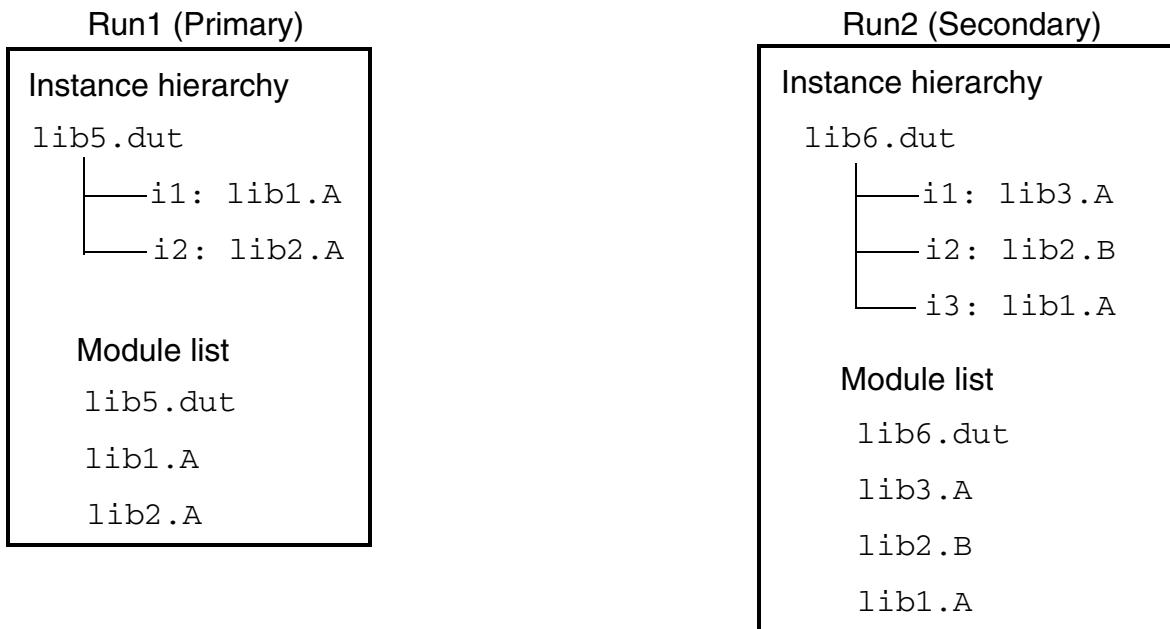
The first `merge_config` command creates `Tb-system.System` at the root of the target model and names it `my_System`. Notice that in the second `merge_config` command, `-targettype` is used. As a result, a new node is not created. The merge operation will assume that `A` already exists in the target model. In this case, `A` exists because of the use of previous `merge_config` command. If the first `merge_config` command was not used, an error would have been reported to indicate that `A` does not exist in the target model. The merge operation will merge `TB_A.A` to each instance of `A` that exists in the target model.

5.2.3.4 Merge Behavior if Modules with Same Name are Compiled in Different Libraries

IMC, by default, allows you to merge runs that include modules with the same name even if they are compiled in different libraries during simulation.

Consider the instance and module hierarchies shown in [Figure 5-26](#) on page 293.

Figure 5-26 Sample Instance and Module Hierarchies



In the given example (in the case of standard merge):

- Instance-level merging will take place based on the hierarchical name of the instance.
- Module-level merging will be as stated below:

- ❑ Module `lib6.dut` from secondary run will be merged to the module `lib5.dut` in the primary run because there is one module with the same name even though the library names are different in the primary run.
- ❑ Module `lib3.A` from the secondary run will not be merged to any of the modules in the primary run, because there is more than one module with the same name, `lib1.A` and `lib2.A`, in the primary run. In this case, a warning is reported.
- ❑ Module `lib2.B` from the secondary run will not be merged because there is no module with the same name in the primary run. In this case, a warning is reported.
- ❑ Module `lib1.A` from the secondary run will be merged to `lib1.A` in the primary run because there is a module with the same name, `lib1.A` in the primary run.

In the given example (in the case of union merge):

- Instance-level merging will take place based on the hierarchical name of the instance.
- Module-level merging will be as stated below:
 - ❑ Module `lib6.dut` from secondary run will be merged to the module `lib5.dut` in the target run because there is one (and only one) module with the same name even though the library names are different in the target run.
 - ❑ Module `lib3.A` will get added to the target run as part of union merge.
 - ❑ Module `lib2.B` will get added to the target run as part of union merge. However, its coverage will be 0 because no matching instance is found in the target run.
 - ❑ Module `lib1.A` from the secondary run will be merged to `lib1.A` in the target run.

Important

If at the time of coverage data generation, the `set_merge_with_libname` command was used, then merge will not be allowed for modules with different library names. For more details on this command, see the *ICC User Guide*.

Note: In the previous releases, by default, the merge was not allowed for runs with differences in library information. To enable merge with different library information, when work libraries were different across runs, it was important that both primary and secondary tests are dumped with the `set_ignore_library_name` command during elaboration. However, with this release, the `set_ignore_library_name` command is made redundant. Therefore, coverage databases that were generated in the default mode before 12.2 release cannot be merged with the databases generated in the default mode with 12.2 release.

Note: Both the commands (`set_merge_with_libname` and `set_ignore_library_name`) are ICC commands. For more details on these commands, see the *ICC User Guide*.

5.3 Precedence while Merging

When merging coverage using the `merge` command, the result is the strongest of the values, as defined here:

COVERED > IGN > UNCOVERED

If you want IGN to be considered the strongest, use:

```
preferences -set merge.ignStrongest true
```

With the above command, the precedence is changed as:

IGN > COVERED > UNCOVERED

For more details on the `preferences` command, see [Setting/Viewing Preferences](#) on page 249.

Incisive Metrics Center User Guide

Ranking Runs

Ranking runs helps you identify the runs that:

- Make the most efficient test suite for regression testing, eliminating the redundant runs.
- Help in achieving the target coverage goal

Ranking the runs involves:

- Setting the Ranking Configuration using the `rank_config` command (optional)
- Ranking the Runs using the `rank` command

Note: Ranking runs does not require loading of runs.

Note: Ranking is a heavy procedure, and is recommended that it is performed over a 64-bit version of IMC.

6.1 Setting the Ranking Configuration

When ranking runs, you can specify the factors to be considered while ranking. These factors include the weight of different coverage types, the maximum runs to rank, and the target to be achieved while ranking. These factors are specified using the `rank_config` command. The `rank_config` command allows you to configure the ranking behavior as required.

6.1.1 Command Syntax

The BNF of the `rank_config` command is:

```
rank_config [metric_type_weight_option]
[-max <integer_value>]
[-self_grades on|off]
[-target <cumulative_coverage_goal>]
[-initial_runfile <filename>]
[-cost cpu|sim]
[-refinement_files <filenames>]
```

```
metric_type_weight_option ::= [-w_block [<weight>]] [-w_expression [<weight>]]  
[-w_toggle [<weight>]] [-w_fsm [<weight>]] [-w_assertion [<weight>]]  
[-w_covergroup [<weight>]]
```

In the above BNF:

- `metric_type_weight_option` increases or decreases the weight (`<weight>`) of a given metric type. The default weight, if not specified, is 1. A value of 1 indicates that all available metric types are considered while ranking. If the weight is 0, then coverage for that item is not considered while ranking runs. In ranking runs, `<weight>` can be specified with:

- `-w_block` for block coverage
 - `-w_expression` for expression coverage
 - `-w_toggle` for toggle coverage
 - `-w_fsm` for FSM state coverage and transition coverage
 - `-w_assertion` for assertion coverage
 - `-w_covergroup` for covergroup coverage

- `-max <integer_value>` specifies the number of runs after which ranking runs must stop. It helps you identify the best N runs.
- `-self_grades on|off` specifies if the self grades must be shown in the rank report or not. By default, the self grades are shown in the rank report. To turn off showing the self grades in the rank report, use the `-self_grades off` option.

Note: Self grade attributes will be deprecated soon. Turning off self grades improves performance. To improve performance, it is recommended to turn off the self grades using the `rank_config -self_grades off` command.

- `-target <cumulative_coverage_goal>` specifies the coverage target goal to be achieved while ranking. The ranking operation stops after the specified coverage target goal is achieved.
- `-initial_runfile <filename>` allows you to specify a run file that contains a list of runs that must be included always in the ranking results. The runs specified in the `<filename>` appear above the runs specified for ranking. Run names in the `<filename>` are separated by new lines (that is, each run name is on a separate line in the file), and can be full run names or contain wildcards.
- `-cost cpu|sim` specifies the cost to be considered while ranking runs. It can be any of following:
 - `cpu` to consider CPU time while ranking runs.
 - `sim` to consider simulation time while ranking runs.

For an example on applying cost while ranking, see [Example: Use of -cost Option](#) on page 310.

- `-refinement_files <filenames>` specifies refinement files to be considered while ranking runs. You can specify more than one refinement file by separating the file names with a comma (,). The entities specified in the refinement files are considered excluded and are ignored while ranking the runs. For an example on applying exclusions while ranking, see [Example: Using Refinement Files During Ranking](#) on page 309.

Examples

To rank runs to achieve a cumulative coverage goal of 90%, use:

```
rank_config -target 90
```

To ignore block and toggle coverage while ranking runs, use:

```
rank_config -w_block 0 -w_toggle 0
```

Note: The configuration specified using the `rank_config` command applies only to the first `rank` command that follows. It does not apply to all subsequent `rank` commands. After the first `rank` command is executed, the configuration settings are automatically reset to default values.

6.2 Ranking the Runs

After specifying the required ranking configuration, use the `rank` command to rank the runs.

6.2.1 Command Syntax

The BNF of the `rank` command is:

```
rank {run_specification | -use_prev_merge}  
[-html | -text]  
[-out_html <html_output_file>]  
[-out_text <text_output_file>]  
[-out <out_path>]  
[-run_order {as_specified | by_timestamp} ]  
[-overwrite]  
[-attach on]  
[rank_element_option]
```

where

```
run_specification ::= {<runs> | -runfile <runfile>}  
rank_element_option ::=  
-entity <path> [-entity_type {type | inst | toggle | fsm | assertion | covergroup  
| coveritem | coverbin}] |
```

```
-type <type> |
-inst <instance> |
-covergroup <cover_group_element_full_path_name> |
-coverpoint <cover_point_element_full_path_name> |
-coverbin <cover_bin_element_full_path_name> |
-assertion <assertion_full_path_name> |
-toggle <toggle_full_path_name> |
-fsm <fsm_full_path_name> |
-elements_file <elements_file>
```

In the above BNF:

- `run_specification` specifies the list of runs that must be ranked. You can specify the run list in any of the following ways:
 - At the command line using the `<runs>` option.
 - In a text file using the `-runfile <runfile>` option. The `<runfile>` is the file that includes the list of runs to be ranked. The runs in `<runfile>` should be listed one per line. You can include comments in `<runfile>` using #.

The runs specified in `<runs>` or `<runfile>` can include just the run name or the complete path, as:

```
[ [<workdir>/]<scope>/]<run>
```

where

- `<workdir>` specifies the root location where all coverage data is stored. It should contain valid filename characters and path separators but cannot contain wildcards. `<workdir>` can use both absolute paths (for example, `/home/me/chip/verification/todaysrev`) and relative paths (for example, `verification/todaysrev`).
- `<scope>` specifies a particular configuration of hardware (DUT). It should contain valid filename characters and cannot contain path separators or wildcards.
- `<run>` specifies the run name for a single simulation. It should contain valid filename characters and can contain wildcards.

- `-run_order` specifies the order in which the runs will be processed during the model union stage of ranking. The run order can be any of the following:
 - `as_specified` to process runs in the order as specified on the command line.
 - `by_timestamp` to process runs in reverse chronological order (that is, the latest run is processed first).

Note: If not specified, the run order is assumed to be `as_specified`.

- `-html` specifies that the ranking results must be reported in an HTML format.

- `-text` specifies that the ranking results must be reported in a text format on the standard output.
- `-out_html <html_out_path>` specifies the location where ranking results in HTML format must be stored. If the `-html` option is used, then, by default, the ranking results are redirected to a file named `rank_html_<timestamp>.html` in the current working directory. You can change the default location using the `-out_html` option.
- `-out_text <text_out_path>` specifies the location where ranking results in text format must be stored. If the `-text` option is used, then by default, the rank report in text format is shown on the standard output. You can redirect the output to a file using the `-out_text` option.
- `-out <out_path>` specifies the location where ranking results must be stored. This option cannot be used with `-out_html` or `-out_text` options. The `-out` option will be deprecated soon. If the `-out` option is used without the `-html` or the `-text` option, then the `-text` option is assumed.
- `-overwrite` enables overwriting of the existing HTML rank output directory. By default, HTML rank output directory is not overwritten and an error is reported. This option is ignored in the case of text reports.
- `-attach on` enables including a link to the HTML ranking output page (`rank.html`) from the HTML reporting output page (`index.html`). To include a link to HTML ranking output page from the HTML reporting output page:
 - Use the `-attach on` option
 - Ensure that the same output directory is specified with the `rank` command and the `report` command.

Note: If `-attach` option is used in combination with the `-overwrite` option, then only the `rank.html` file is overwritten. In addition, you must use the `-attach` option after the HTML report has been generated with the `report -html` command. For more details on using the `-attach` option, see [Example: Use of -attach option](#) on page 313.

- `rank_element_option` specifies the elements for which coverage grading is calculated during ranking. By default, ranking is done for the entire model. To consider only specific elements during ranking, use the `rank_element_option` as:
 - `-entity <path>` to specify the path of the model element targeted for ranking. The path can be prefixed with "*Verification\ Metrics*", "*Verification\ Metrics/Instances*", "*Verification\ Metrics/Types*", "*Instances*", or "*Types*". Prefix is not mandatory. However, if the *Instances* or *Types* prefix is specified, the performance of the `rank` command can be improved. The `-entity_type` option can be specified with the `-entity` option. Valid values for entity type are: `type`, `inst`, `fsm`, `assertion`, `covergroup`, `coveritem`, `coverbin`, and `toggle`. If

entity type is not specified, the `rank` operation will search for the entities under all metric options and it can be an expensive operation. Using the `-entity_type` option can improve the performance of the `rank` command.

Note: Wild cards (*, ?, and . . .) are not supported in the prefix *Verification\ Metrics/Types* or *Verification\ Metrics/Instances*.

- ❑ `-type <type>` to specify the module for which coverage grading must be calculated during ranking.
- ❑ `-inst <instance>` to specify the instance for which coverage grading must be calculated during ranking.
- ❑ `-covergroup <cover_group_element_full_path_name>` to specify the covergroups for which coverage grading must be calculated during ranking.
- ❑ `-coverpoint <cover_point_element_full_path_name>` to specify the coverpoints for which coverage grading must be calculated during ranking.
- ❑ `-coverbin <cover_bin_element_full_path_name>` to specify the coverbins for which coverage grading must be calculated during ranking.
- ❑ `-assertion <assertion_full_path_name>` to specify the assertions for which coverage grading must be calculated during ranking.
- ❑ `-toggle <toggle_full_path_name>` to specify the toggles for which coverage grading must be calculated during ranking.
- ❑ `-fsm <fsm_full_path_name>` to specify the FSMs for which coverage grading must be calculated during ranking.
- ❑ `-elements_file <elements_file>` to specify multiple entities for ranking. You can specify the entities to be ranked in a file `<elements_file>` and then pass this file as an argument to the `rank` command. The number of entities specified in the `<elements_file>` should not exceed 1000.

For example, you can specify the following entities in a file named `rankelements` and then pass this file as an argument to the `rank` command:

```
-type mux41
-inst DUT/u2
-covergroup DUT/u5/cg_inst
-coverpoint DUT/u5/cg_inst/FSM*
-assertion DUT/u5/MST_ST*
-inst -covergroup DUT/u5/cg_inst
-type -assertion ctrl_path2/FSM2*
```

You can then pass this file to the `rank` command as:

```
rank test1 test2 test3 test4 -elements_file rankelements
```

Note: The syntax rules for specifying instance names and type names are same as the ones for the `exclude` command.

- `-use_prev_merge` helps you to rank the runs that were merged earlier. This option is useful if you want to perform ranking after doing a merge. For example, you add runs, and then merge the runs to determine if your coverage goal has been met. In addition, you also want to simultaneously check if the newly added run was redundant. The `-use_prev_merge` option is useful in such scenarios.

Note: This option works only if a merge operation is performed before using the `rank` command. For details on merging runs, see [Merging Runs](#) on page 259. In addition, this option (`-use_prev_merge`) is not supported in the case of parallel ranking. For more details on parallel ranking, see [Configuring Parallel Ranking Options](#) on page 99.

For an example, see [Example with Previous Merge Option](#) on page 308.

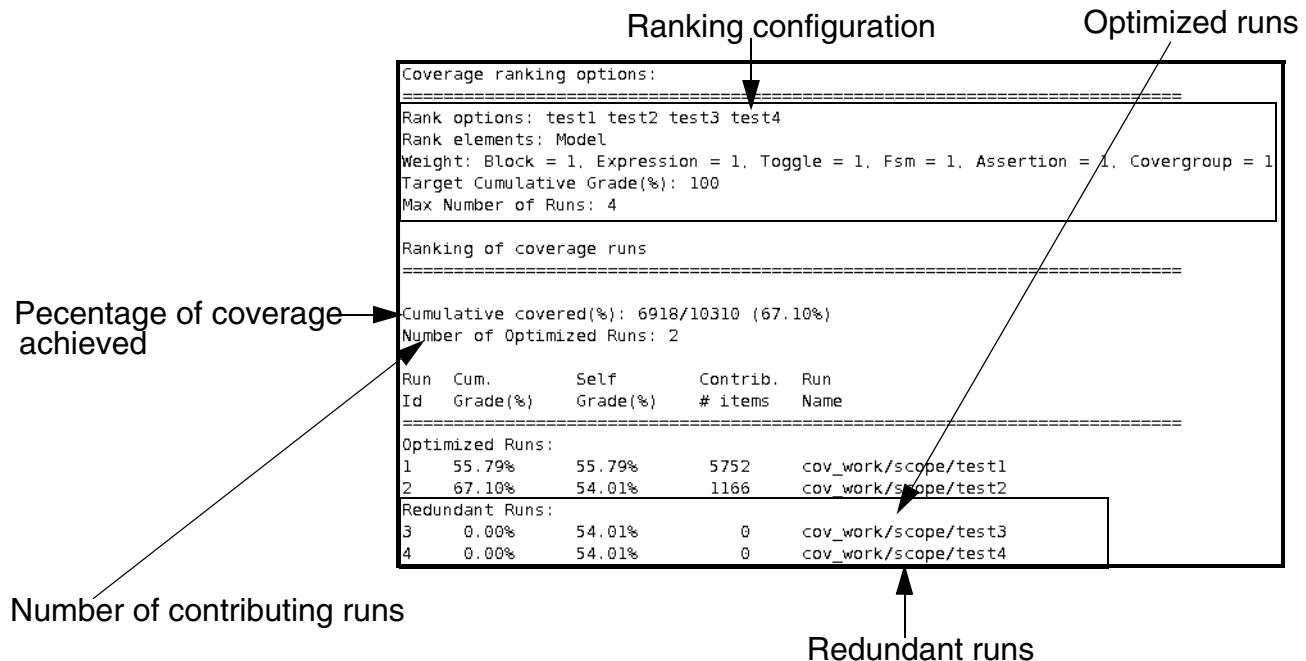
When ranking runs, remember that:

- The `rank` action does not support functional coverage attributes `weight`, `goal`, and `at_least`. These attributes are not considered while ranking.
- The tool is now running multi-thread upon `rank` action. By default, the number of threads is set as 8 and should not be changed unless absolutely required. However, if required for any reason, you can modify it. For this, open `.mdv/internal.properties` from the home directory and then edit the `application.threadpool.rank-thread-pool.max_pool_size` to the required size.

6.2.2 Ranking Examples (Text Format)

[Figure 6-1](#) on page 304 shows a sample output of a `rank` command.

Figure 6-1 Output of Rank Command



In the report, the best run is listed first. The run listed first has the best results according to coverage grading. Additional tests are listed in descending order of that value. The above results indicate that `test3` and `test4` are redundant, and `test1` contributes the maximum to overall coverage results.

In the above report:

Coverage ranking options	Displays the rank configuration information. The above report displays default values because the <code>rank_config</code> command was not used before ranking the runs.
Optimized Runs	Lists the runs that contribute to overall coverage grade.
Redundant Runs	Lists the runs that do not contribute to overall coverage grade.
Run Id	Is the unique ID assigned sequentially to identify the coverage run. Indexes start at 1 and increase by one for each additional coverage run.
Cum Grade (%)	Is the cumulative coverage grade for the run.
Self Grade (%)	Is the coverage grade of the run.

Contrib. # items	Shows the total number of coverage items (of the enabled metrics) of that particular run contributing to the overall coverage grade. For example, in Figure 6-1 on page 304, test1 contributes 5752 coverage items to the overall coverage grade and test2 contributes additional 1166 coverage items to the overall coverage grade. For redundant runs, this column displays a 0 because there are no additional items that contribute to the coverage grade.
Run Name	Is the name of the run.

6.2.2.1 Examples

To rank runs considering only the coverage grade of module m1, use:

```
rank test1 test2 test3 test4 -type m1
```

To rank runs to achieve a cumulative coverage goal of 90%, use:

```
rank_config -target 90  
rank test1 test2 test3 test4
```

To ignore block and expression coverage while ranking runs, use:

```
rank_config -w_block 0 -w_expression 0  
rank test1 test2 test3 test4
```

To output the result of ranking operation to a file named rankedruns, use:

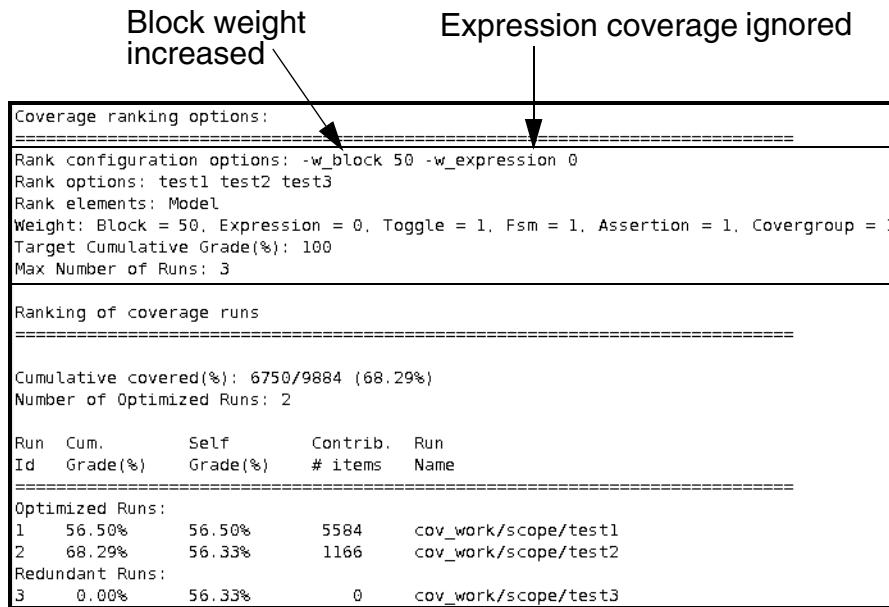
```
rank test1 test2 test3 test4 -out rankedruns
```

To ignore expression coverage, and increase the weight of block coverage by 50 while ranking the runs, use:

```
rank_config -w_expression 0 -w_block 50  
rank test1 test2 test3
```

[Figure 6-2](#) on page 306 shows output of above set of commands.

Figure 6-2 Output of Rank Command



Coverage ranking options:
=====

Rank configuration options: -w_block 50 -w_expression 0
Rank options: test1 test2 test3
Rank elements: Model
Weight: Block = 50, Expression = 0, Toggle = 1, Fsm = 1, Assertion = 1, Covergroup = 1
Target Cumulative Grade(%): 100
Max Number of Runs: 3

Ranking of coverage runs
=====

Cumulative covered(%): 6750/9884 (68.29%)
Number of Optimized Runs: 2

Run Id	Cum. Grade(%)	Self Grade(%)	Contrib. # items	Run Name
1	56.50%	56.50%	5584	cov_work/scope/test1
2	68.29%	56.33%	1166	cov_work/scope/test2
3	0.00%	56.33%	0	cov_work/scope/test3

Notice that the weight option set while ranking appear in the Coverage ranking options section of the report.

To turn off showing of self grades from the rank report, use:

```
rank_config -self_grades off  
rank test1 test2 test3 test4
```

Figure 6-3 on page 307 shows output of above set of commands.

Figure 6-3 Output of Rank Command

```
Coverage ranking options:
=====
Rank configuration options: -self_grades off
Rank options: test1 test2 test3 test4
Rank elements: Model
Weight: Block = 1, Expression = 1, Toggle = 1, Fsm = 1, Assertion = 1, Covergroup = 1
Target Cumulative Grade(%): 100
Max Number of Runs: 4

Ranking of coverage runs
=====
Cumulative covered(%): 6918/10310 (67.10%)
Number of Optimized Runs: 2

Run  Cum.      Contrib.  Run
Id   Grade(%)   # items   Name
=====
Optimized Runs:
1    55.79%     5752    cov_work/scope/test1
2    67.10%     1166    cov_work/scope/test2
Redundant Runs:
3    0.00%      0       cov_work/scope/test3
4    0.00%      0       cov_work/scope/test4
```

Self grade not shown

Notice that self grade is not shown in the report now.

To definitely include runs specified in alwaysrank.txt, use:

```
rank_config -initial_runfile alwaysrank.txt
rank test1 test2 test4
```

Figure 6-4 on page 308 shows output of the above set of commands.

Figure 6-4 Output of Rank Command

```
Coverage ranking options:
=====
Rank configuration options: -initial_runfile alwaysrank.txt
Rank options: test1 test2 test4
Rank elements: Model
Weight: Block = 1, Expression = 1, Toggle = 1, Fsm = 1, Assertion = 1, Covergroup = 1
Target Cumulative Grade(%): 100
Max Number of Runs: 4

Ranking of coverage runs
=====

Cumulative covered(%): 6918/10310 (67.10%)
Number of Optimized Runs: 1

Run Cum. Self Contrib. Run
Id Grade(%) Grade(%) # items Name
=====
Initial Runs:
1 54.01% 54.01% 5568 ./cov_scope/test3
Optimized Runs:
2 67.10% 55.79% 1350 cov_scope/test1
Redundant Runs:
3 0.00% 54.01% 0 cov_scope/test2
4 0.00% 54.01% 0 cov_scope/test4
```

Initial runs
(specified in the initial runfile)



The above output shows initial runs specified in the `alwaysrank.txt` file in the Initial Runs section of the report.

Example with Previous Merge Option

The following rank operation uses the `-use_prev_merge` option.

```
merge test1 test2 test3 test4 -out merged_result
rank -use_prev_merge
```

[Figure 6-5 on page 309](#) shows output of the above set of commands.

Figure 6-5 Output of Rank Command

Indicates model used of previous merge operation

```

imc> rank -use_prev_merge
*W_SLFGRD: Warning: Self grade attributes will be deprecated soon. To improve performance, use the rank_config -self_grades off option. This will turn OFF showing the self grade attributes in the ranking report.
Using resultant model of previous merge . . .
*W_FWALG: Warning: Rank action does not support functional coverage attributes weight/goal/at_least. Such attributes are not considered while ranking and as a result ranking grades might not be aligned with those presented in context.
*W_SRCALC: Reading and processing of selected runs for rank is done, starting actual ranking...
Coverage ranking options:
=====
Rank options: -use_prev_merge
Rank elements: Model
Weight: Block = 1, Expression = 1, Toggle = 1, Fsm = 1, Assertion = 1, Covergroup = 1
Target Cumulative Grade(%): 100
Max Number of Runs: 4

Ranking of coverage runs
=====

Cumulative covered(%): 6918/10282 (67.28%)
Number of Optimized Runs: 2

Run  Cum.  Self  Contrib.  Run
Id  Grade(%)  Grade(%)  # items  Name
=====
Optimized Runs:
1    55.94%  55.94%    5752  cov_work/scope/test1
2    67.28%  54.15%   1166  cov_work/scope/test2
Redundant Runs:
3    0.00%  54.15%     0  cov_work/scope/test3
4    0.00%  54.15%     0  cov_work/scope/test4

```

In the above output, notice that the resultant model of the previous merge operation is used during ranking. This reduces the time taken to rank the runs.

Note: You might observe a few differences in IMC ranking as compared with ICCR ranking. See [Differences from ICCR Functionality](#) on page 448 for more details.

Example: Using Refinement Files During Ranking

Consider the following rank commands:

```

rank_config -w_block 1 -w_expression 0 -w_toggle 0 -w_fsm 0 -w_assertion 0
-w_covergroup 0
rank TB1 TB4 -inst DUT.u5

```

[Figure 6-6 on page 309](#) shows output of the above set of commands.

Figure 6-6 Output of Rank Command

Run	Cum.	Self	Contrib.	Run
Id	Grade(%)	Grade(%)	# items	Name
=====				
Optimized Runs:				
1	79.71	79.71	55	cov_work/scope/TB1
2	84.06	79.71	3	cov_work/scope/TB4

In the above report, TB1 contributes 55 items and TB4 contributes 3 item to the ranking results.

Refinement file `b1.vRefine` includes refinements rules for 2 blocks that were hit within TB1 and not within TB4. If we apply this refinement file during ranking, the rank results will change. To apply this refinement file at the time of ranking, use the following commands:

```
rank_config -refinement_files b1.vRefine -w_block 1 -w_expression 0 -w_toggle 0  
-w_fsm 0 -w_assertion 0 -w_covergroup 0  
rank TB1 TB4 -inst DUT.u5
```

[Figure 6-7 on page 310](#) shows output of the above set of commands.

Figure 6-7 Output of Rank Command

Run	Cum.	Self	Contrib.	Run
Id	Grade(%)	Grade(%)	# items	Name
<hr/>				
Optimized Runs:				
1	82.09	82.09	55	cov_work/scope/TB4
2	83.58	79.10	1	cov_work/scope/TB1

In the above report, TB4 becomes the new best run. This is due to the refinements applied during ranking. The entities specified in the refinement files are considered excluded and are ignored while ranking the runs.

Example: Use of `-cost` Option

Consider the following `rank` command:

```
rank test1 test2 test3 test4
```

[Figure 6-8 on page 310](#) shows output of the above set of command.

Figure 6-8 Output of Rank Command

Ranking of coverage runs				
<hr/>				
Cumulative covered(%): 6918/10310 (67.10%)				
Number of Optimized Runs: 2				
Run	Cum.	Self	Contrib.	Run
Id	Grade(%)	Grade(%)	# items	Name
<hr/>				
Optimized Runs:				
1	55.79%	55.79%	5752	cov_work/scope/test1
2	67.10%	54.01%	1166	cov_work/scope/test2
Redundant Runs:				
3	0.00%	54.01%	0	cov_work/scope/test3
4	0.00%	54.01%	0	cov_work/scope/test4

If you want to consider simulation time while ranking the runs, use the following commands:

```
rank_config -cost sim
```

```
rank test1 test2 test3 test4
```

Figure 6-9 on page 311 shows output of the above set of commands.

Figure 6-9 Output of Rank Command

New column added

Ranking of coverage runs					
=====					
Cumulative covered(%): 6918/10310 (67.10%)					
Number of Optimized Runs: 2					
Run Id	Cum. Grade(%)	Self Grade(%)	Contrib. # items	Cost	Run Name
=====					
Optimized Runs:					
1	55.79%	55.79%	5752	2126000	cov_work/scope/test1
2	67.10%	54.01%	1166	3164600	cov_work/scope/test2
Redundant Runs:					
3	0.00%	54.01%	0	3164600	cov_work/scope/test3
4	0.00%	54.01%	0	3164600	cov_work/scope/test4

In the above report, a new column Cost is added.

IMC uses the following formula while ordering the runs:

Contribution / Cost

For example, consider the following list of runs:

Run	Contribution	Cost (cpu/sim)
Run1	100	500
Run2	80	200

If simulation time is not considered while ranking, then Run1 is ranked first. However, if simulation time is considered while ranking, then Run2 becomes the best run and is ranked first because 80/200 is more than 100/500.

6.2.3 Ranking Examples (HTML Format)

To generate ranking results in an HTML format and to send the output to directory named htmlrank1, use:

```
rank test1 test2 test3 test4 -html -out htmlrank1 -inst *...
```

Incisive Metrics Center User Guide

The above command creates a directory names `htmlrank1` in the current working directory and creates following files and folders in it.

File/Directory Name	Description
<code>rank_sub_dir</code>	This directory includes a file <code>rank.html</code> . This file includes ranking results in an HTML format. You must open this file to view the ranking results in an HTML format.
<code>rankLegend.html</code>	This is the help page of HTML rank report. It shows the detailed description of various columns and items shown in the HTML rank report.

To view the ranking results generated with the above command, open `htmlrank1/rank_sub_dir/rank.html` in a standard web browser.

[Figure 6-10](#) on page 312 shows a sample output of a `rank` results in an HTML format.

Figure 6-10 Output of Rank Command in HTML Format

Ranking configuration

Click to view legend and help

Ranking results

IMC Ranking Report - Mozilla Firefox

Coverage ranking options

Rank options: test1 test2 test3 test4 -inst *... -html -out htmlrank1

Rank Elements: Model

Weight: Block = 1, Expression = 1, Toggle = 1, Fsm = 1, Assertion = 1, Covergroup = 1

Target Cumulative Grade(%): 100

Max Number of Runs: 4

Ranking of coverage runs

Cumulative covered(%): 6918/10310 (67.10%)

Number of Optimized Runs: 2

Optimized Runs:

Run Id	Cum Grade(%)	Self Grade(%)	Contrib. # Items	Run Name
1	55.79%	55.79%	5752	cov_work/scope /test1
2	67.10%	54.01%	1166	cov_work/scope /test2

Redundant Runs:

Run Id	Cum Grade(%)	Self Grade(%)	Contrib. # Items	Run Name
3	0.00%	54.01%	0	cov_work/scope /test3
4	0.00%	54.01%	0	cov_work/scope /test4

Coverage Color Legend

0	<25	<50	<75	<100	100	n/a	Not Scored
---	-----	-----	-----	------	-----	-----	------------

Done

The above results indicate that `test3` and `test4` are redundant, and `test1` contributes the maximum to overall coverage results.

Example: Use of -attach option

In case you want that the ranking results generated in an HTML format are also available as a link from the HTML report, you must:

- Use the `-attach on` option with the `rank` command, and
- Ensure that the HTML output directory specified with the `rank` command is same as the HTML output directory specified with the `report` command.

Consider the following set of commands:

```
report -html -out html_rep1 -inst *...  
rank test1 test2 test3 test4 -html -out html_rep1 -attach on -inst *...
```

In the above commands:

- Same output directory is specified with both the commands.
- The `-attach` option is turned on.

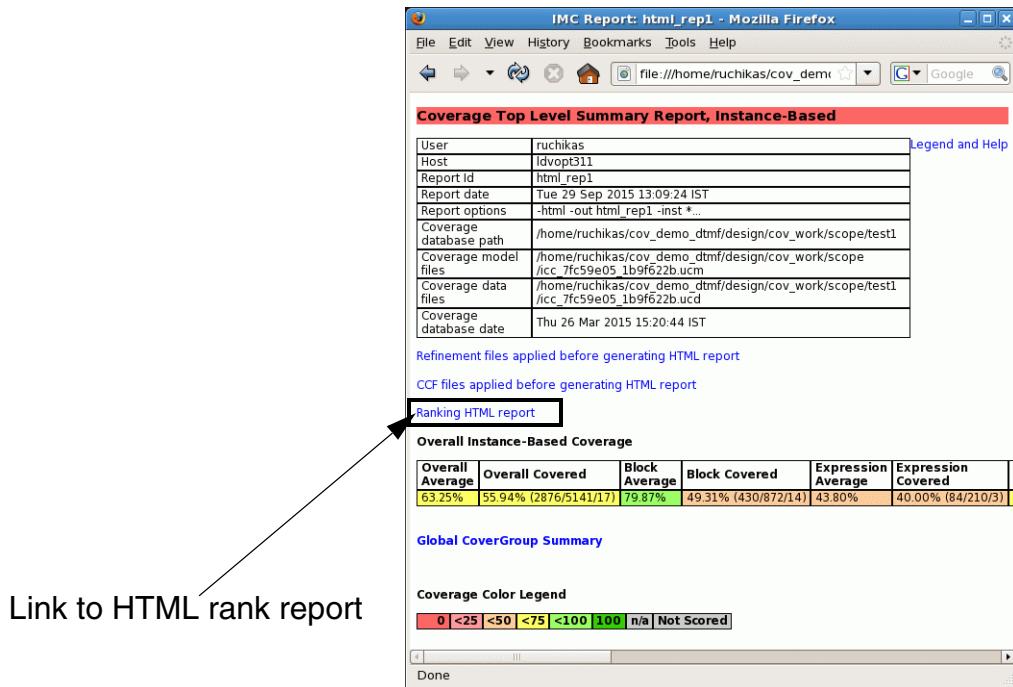
The above commands create a directory named `html_rep1` in the current working directory and following files and folders are created in it.

File/Directory Name	Description
<code>rank_sub_dir</code>	This directory includes a file <code>rank.html</code> . This file includes ranking results in an HTML format.
<code>report_sub_dir</code>	See Generating Reports in an HTML Format Using report -html Command on page 371 for details on this directory.
<code>list_refinement</code>	See Generating Reports in an HTML Format Using report -html Command on page 371 for details on this directory.
<code>global_cg_summ.html</code>	See Generating Reports in an HTML Format Using report -html Command on page 371 for details on this directory.
<code>legend.html</code>	See Generating Reports in an HTML Format Using report -html Command on page 371 for details on this directory.

File/Directory Name	Description
index.html	It is the top-level summary page using which you can navigate through the HTML reports. You must open this page in the standard web browser to navigate through the HTML reports. As the <code>-attach</code> option is used in this example, this page also includes a link to the HTML ranking results.
rankLegend.html	This is the help page of HTML rank report. It shows the detailed description of various columns and items shown in the HTML rank report.

Figure 6-11 on page 314 shows the HTML report with a link to HTML rank report.

Figure 6-11 Link to HTML Rank Report from HTML Report



The above top-level summary page of the HTML report now includes a link to the HTML rank report. You can click this link to view the HTML ranking results.

For details on generating HTML reports, see [Generating Reports in an HTML Format Using report -html Command](#) on page 371.

Reporting and Grading

This chapter discusses how to generate coverage reports in the batch mode as well as GUI mode of IMC and also discusses the various grading schemes used in grade calculations.

This chapter discusses the following topics:

- [Generating Reports in GUI](#) on page 315
- [Generating Reports in Interactive Command-line Mode](#) on page 327
- [Coverage Grading in IMC](#) on page 407

7.1 Generating Reports in GUI

In GUI, you can generate reports from top-level *Metrics* page and also for individual detailed pages.

This section discusses generating reports at the top-level that is from the *Metrics* page. For information on generating reports for individual detailed pages, see [Chapter 2, “Analyzing Metrics Data.”](#)

Generating Reports from top-level Metrics page

The *Report* toolbar has a drop-down button named *Report* using which you can generate reports from GUI. The *Reports* option is also available in the *Analysis* menu using which you can generate reports.

Note: The report action considers the configuration option *Show extend metrics tree* while generating the report. If the *Show extend metrics tree* option is enabled (using the *Configuration* dialog box in GUI or CLI `config` command), then the covergroups, cover items, FSMs, and assertions are also listed in the metrics hierarchy tree in the report.

The *Report* drop-down button on the *Report* toolbar has the following options for generating reports:

- All (Default)—To generate a metrics report for the complete hierarchy.
- On Selected Entity—To generate a sessions report for a selected entity.

Note: Only HTML reports can be generated from IMC GUI. You cannot generate ASCII reports from GUI.

Consider an example of generating a report for the complete hierarchy.

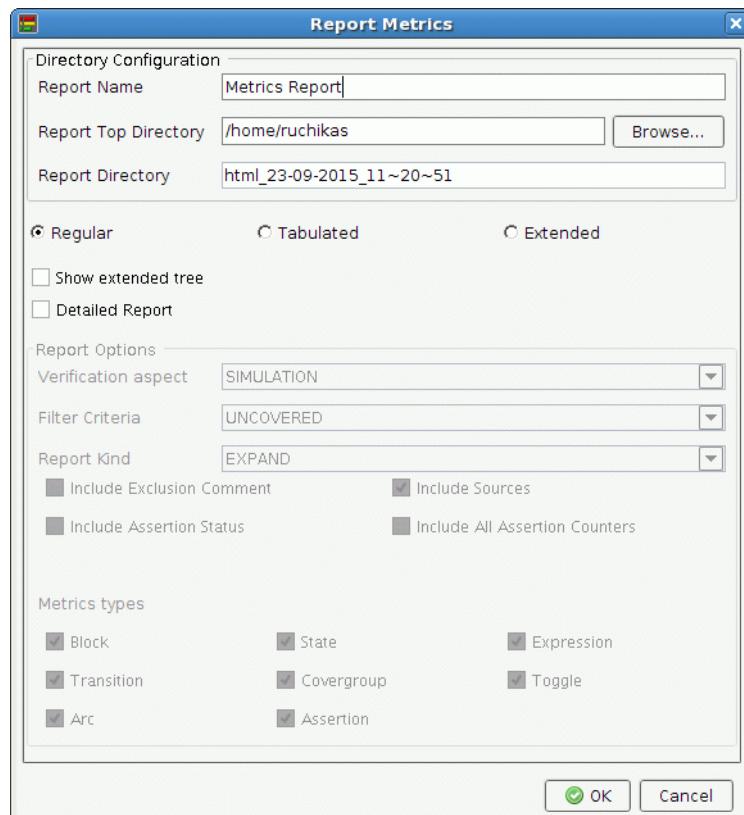
To generate a report for the complete hierarchy:

1. Click the down-arrow next to the *Report* button in the *Report* toolbar and select *All (Default)*.

Note: Alternatively, from the *Analysis* menu, you can select *Reports* —> *Report*.

The *Report Metrics* dialog box is displayed, as shown in [Figure 7-1](#) on page 316.

Figure 7-1 Report Metrics



Note: For generating a report on a specific entity, select the entity in the verification hierarchy tree and select *Report on Selected Entity* from the *Report* drop-down

menu. The report will be generated only for the selected entity and not the entire hierarchy.

2. Specify the name of the report in the *Report Name* text box. By default, the name is specified as *Metrics Report*. This name appears on the HTML report.
3. In the *Report Top Directory* field, specify the location where the report must be created. By default, it shows the location of the current working directory.
4. Specify the name of the report directory in the *Report Directory* text box. By default, it shows the name as `html_<timestamp>`. For example, change the name as `report_all_details`.

Note: When you generate a report from GUI, a directory named `html_<timestamp>` or the name you specified, is created in the current working directory.

5. Specify the type of report to be generated. It can be any of the following:
 - Regular: This is the default report. It lists nodes and sub-nodes and provides hyperlinks for navigating further. For more details, see [Viewing Reports Generated from IMC GUI](#) on page 319.
 - Tabulated: This option generates a bin-level tabulated report. This report shows *Metrics* tree and bin details in tables side-by-side. For more details, see [Example: Bin-Level Tabulated Report](#) on page 325.
 - Extended: This option generates an extended report. In this report, the *Metrics* tree and bins information for each of the cover items is shown in the tree. For more details, see [Example: Bin-Level Extended Report](#) on page 326.
6. Select the *Show extended tree* check box to list covergroups, cover items, FSMs, and assertions along with the types and instances in the hierarchy tree. If you want to list only the types and instances in the hierarchy tree, then clear this check box.
7. To generate a detailed report, select the *Detailed Report* check box. This enables the *Report Options* section of the *Report Metrics* dialog box. By default, this check box is not selected.
8. In the *Verification aspect* drop-down list, specify the assertion properties to be shown in the report.
 - Simulation** shows only simulation assertion properties in the report.
 - Formal** shows only formal assertion properties in the report.
 - Both** shows both simulation and formal assertion properties in the report.

Note: By default, the value is set to **Simulation**.

9. You can specify the items that should be included in the report from the *Filter Criteria* drop-down list. Filter criteria can be *Covered*, *Uncovered*, *Both* (covered and uncovered) *Excludes*, or *All*. For example, select *All* to report all items.
10. You can specify the kind of report in the *Report Kind* drop-down list. Report kind can be *Expand* (prints bin information as a list) or *Aggregate* (prints the cross tuple information in a tabular format after grouping similar bins).
11. By default, the *Include Exclusion Comment* check box is not selected. This means that in the detailed report, the exclusion comments and reviewer information will not be shown. In case, you want to show this information in the report, select the *Include Exclusion Comment* check box.
12. By default, the *Include Sources* check box is selected. This means that in the detailed report, the source information will be shown in the *Source code* column. In case, you want to omit this column in the report, deselect the *Include Sources* check box.

Note: In case you have entities from the compressed files, then even if the *Include Sources* check box is selected, the *Source Code* column will be blank for entities from compressed files.
13. By default, the *Include Assertion Status* check box is not selected. This means that in the detailed report, the *Status* column, which shows the status of the assertion will not be shown in the report. In case, you want to show this information in the report, select the *Include Assertion Status* check box.
14. By default, the *Include All Assertion Counters* check box is not selected. This means that in the detailed report, the *Vacuous*, *Attempt*, and *Disabled* counters will not be shown. In case, you want to show these counters in the report, select the *Include All Assertion Counters* check box.

Note: The *Vacuous*, *Attempt*, and *Disabled* counters are reported only if the `-abvrecordvacuous` switch is used at the time of simulation run. In case the `-abvrecordvacuous` switch is not used at simulation, then the *Vacuous*, *Attempt*, and *Disabled* columns will show *n/a*. For more details on the `-abvrecordvacuous` switch, see the *Verilog Compilation Command-Line Options* user guide.
15. From the *Metrics types* section, select the metrics types for which you want to generate report. To omit a metrics type, clear the respective check box.
16. Click *OK* to generate the report.

The *Report Done* dialog box is displayed, as shown in [Figure 7-2](#) on page 319.

Figure 7-2 Report Done



This dialog box shows the location where the report is generated.

17. Click OK.

You can navigate through the location shown above, and then open the `index.html` file in a Web browser. The `index.html` is the top-level page using which you can view and also navigate through the HTML reports.

Similarly, you can generate more reports based on your requirements.

Viewing Reports Generated from IMC GUI

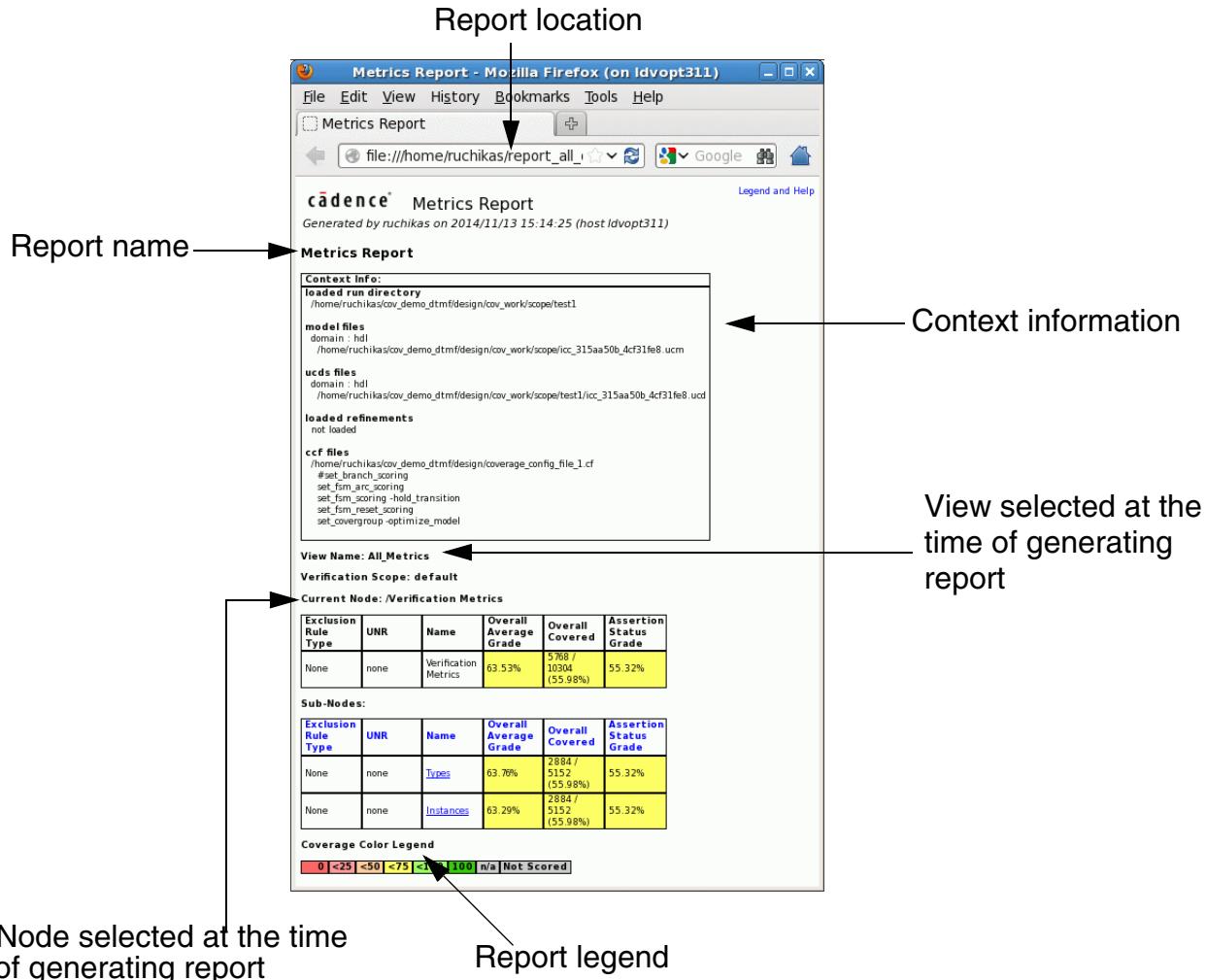
You can view the report by opening the `index.html` file in a Web browser.

To view the report generated earlier using IMC GUI, perform the following steps:

1. Open the `index.html` file in a Web browser.

Figure 7-3 on page 320 shows the top-level page of a regular report generated from IMC GUI.

Figure 7-3 Viewing Regular Report



The top-level page shows information such as:

- ❑ Who generated the report and when the report was generated.
- ❑ The context information, such as coverage database location, model file (.ucm) and data file (.ucd) used to generate the report, the location and contents of the coverage configuration file, and the list of refinement files applied before generating the report.
- ❑ The view used to generate the report. In this case, the default view, *All_Metrics* was selected.

Note: The report output and the columns shown in the report are based on the view selected at the time of generating the report. For details on columns shown in case of different views, see [Views](#) on page 29.

- ❑ The node selected at the time of generating the report.
- ❑ The page also shows links for viewing a list of available types or instances along with the coverage grades.
- ❑ The columns shown in the table are based on the attributes that are shown in the *Verification Hierarchy* pane of the *Summary Metrics* page at the time of generating the report. In this case, it shows the *Overall Average Grade* and *Overall Covered* columns.

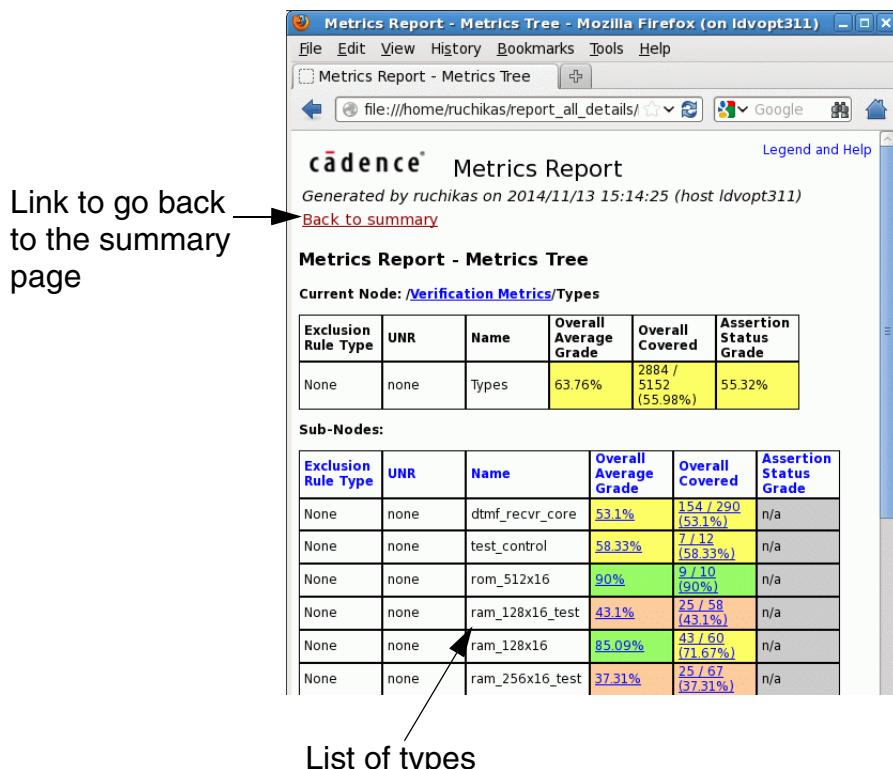
Note: You can adjust the columns that must be included in the coverage report by adding or removing the attributes in GUI and save it as a view before generating the report.

Note: You can sort data in the table by clicking a column header.

2. To navigate through the available types, click the link named *Types*.

Figure 7-4 on page 321 shows list of types along with the coverage grades.

Figure 7-4 Viewing Regular Report

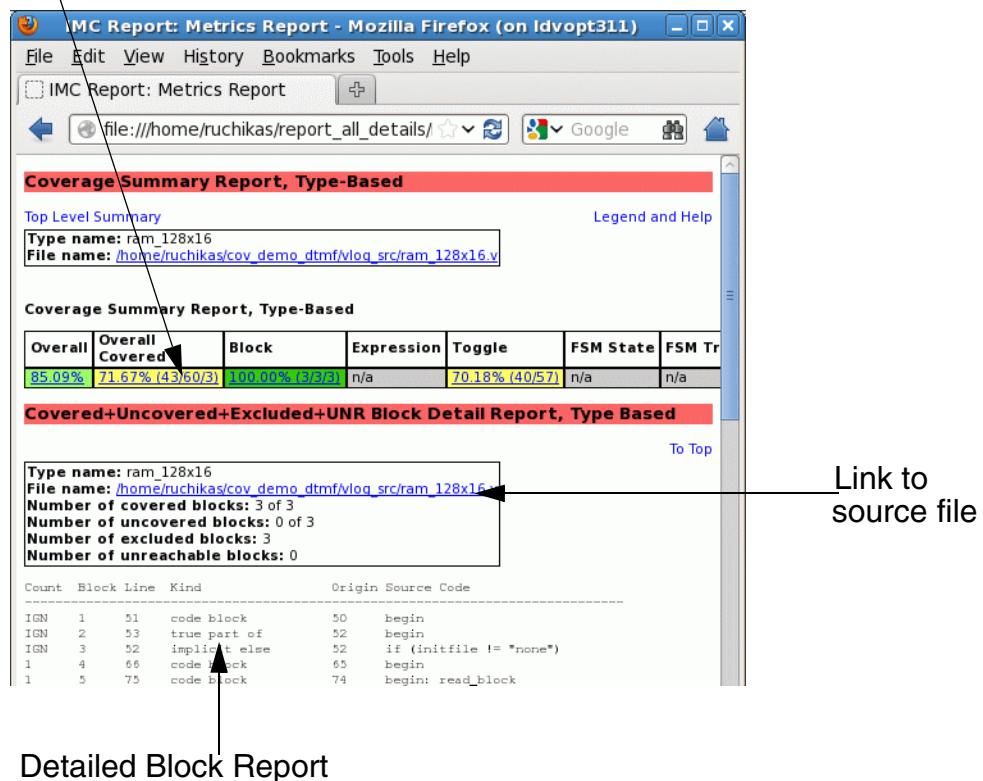


3. You can view the detailed report of any of the types by clicking the hyperlink in the grade columns.

Figure 7-5 on page 322 displays the detailed report.

Figure 7-5 Detailed Report

Summary report showing coverage numbers as links to detailed reports



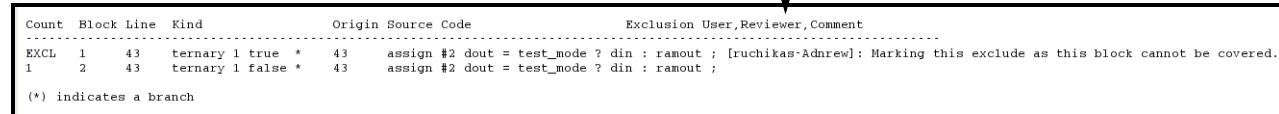
The above report displays the detailed block coverage report of `ram_128x16_test`.

The detailed block coverage report shows information, such as the hit count, the line number, the kind of block, and the source code.

Note: If the *Exclude Comment* check box is selected in the *Report Metrics* dialog box, then an additional column named *Exclusion User*, *Reviewer*, *Comment* is printed in the detailed report, as shown in Figure 7-6 on page 323.

Figure 7-6 Detailed Report

This column is shown if the *Exclude Comments* check box is selected



Count	Block	Line	Kind	Origin	Source Code	Exclusion User,Reviewer,Comment
EXCL 1		43	ternary 1 true *	43	assign #2 dout = test_mode ? din : ramout ;	[truchikas-Adhrew]: Marking this exclude as this block cannot be covered.
1	2	43	ternary 1 false *	43	assign #2 dout = test_mode ? din : ramout ;	

(*) indicates a branch

The values in this column are displayed as:

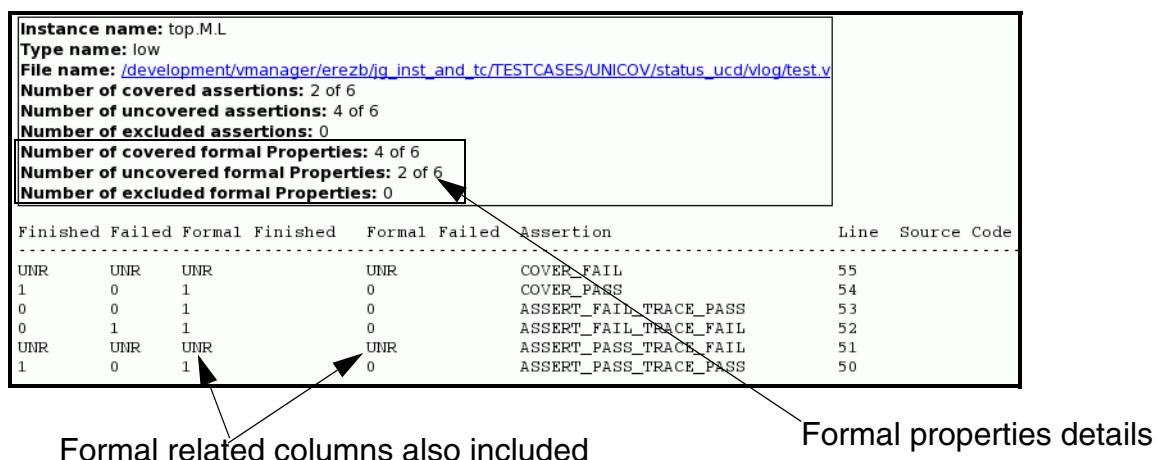
[<user>-<reviewer>] : <comment>

where

- <user> is the login ID of the user.
- <reviewer> is the name of the exclusion reviewer. The exclusion reviewer is specified using the `-reviewer` option of the `exclude` command or in the *Exclusion* dialog box of IMC GUI. If no reviewer is specified at the time of exclusion, the default value `unknown` is considered as the reviewer. For more details, see [Refining Metrics Data](#) on page 167.
- <comment> is the exclusion comment. The exclusion comment is specified using the `-comment` option of the `exclude` command or in the *Exclusion* dialog box of IMC GUI. For more details, see [Refining Metrics Data](#) on page 167.

Note: If the *Verification Aspect* was specified as **BOTH** in the *Report Metrics* dialog box, then additional columns `Formal Finished` and `Formal Failed` are also printed in the detailed report (along with the simulation related columns --> `Finished` and `Failed`), as shown in [Figure 7-7](#) on page 323.

Figure 7-7 Detailed Report (Formal Properties)



Instance name: top.M.L Type name: low File name: /development/vmanager/erezb/ig_inst_and_tc/TESTCASES/UNICOV/status_ucd/vlog/test.v Number of covered assertions: 2 of 6 Number of uncovered assertions: 4 of 6 Number of excluded assertions: 0					
Number of covered formal Properties: 4 of 6 Number of uncovered formal Properties: 2 of 6 Number of excluded formal Properties: 0					
Finished	Failed	Formal	Finished	Formal	Failed
UNR	UNR	UNR	UNR	COVER_FAIL	55
1	0	1	0	COVER_PASS	54
0	0	1	0	ASSERT_FAIL_TRACE_PASS	53
0	1	1	0	ASSERT_FAIL_TRACE_FAIL	52
UNR	UNR	UNR	UNR	ASSERT_PASS_TRACE_FAIL	51
1	0	1	0	ASSERT_PASS_TRACE_PASS	50

Formal related columns also included

Formal properties details

Note: The ungradable items are shown as UNG in the report, as shown in [Figure 7-8](#) on page 324.

Figure 7-8 Detailed Report (Ungradable Items)

Items marked ungradable										
Covergroup Details: c1										
Average Grade	Covered Grade	Goal	Weight	Uncovered Bins	Excluded Bins	Total Bins	Item	Name	Comment	
UNG	UNG	100%	1	0	500625	0	CoverPoint	c1_A		
8%	7% (43/570)	100%	1	527	0	570	CoverPoint	c1_B		

Similarly, you can generate more reports from IMC GUI and also navigate through various reports in any Web browser.

Note: If the *Show extend metrics tree* option is enabled (using the *Configuration* dialog box in GUI or CLI *config* command), then the covergroups, cover items, FSMs, and assertions are also listed in the metrics hierarchy tree, as shown in [Figure 7-9](#) on page 324.

Figure 7-9 Metrics Report (*Show extend metrics tree* option enabled)

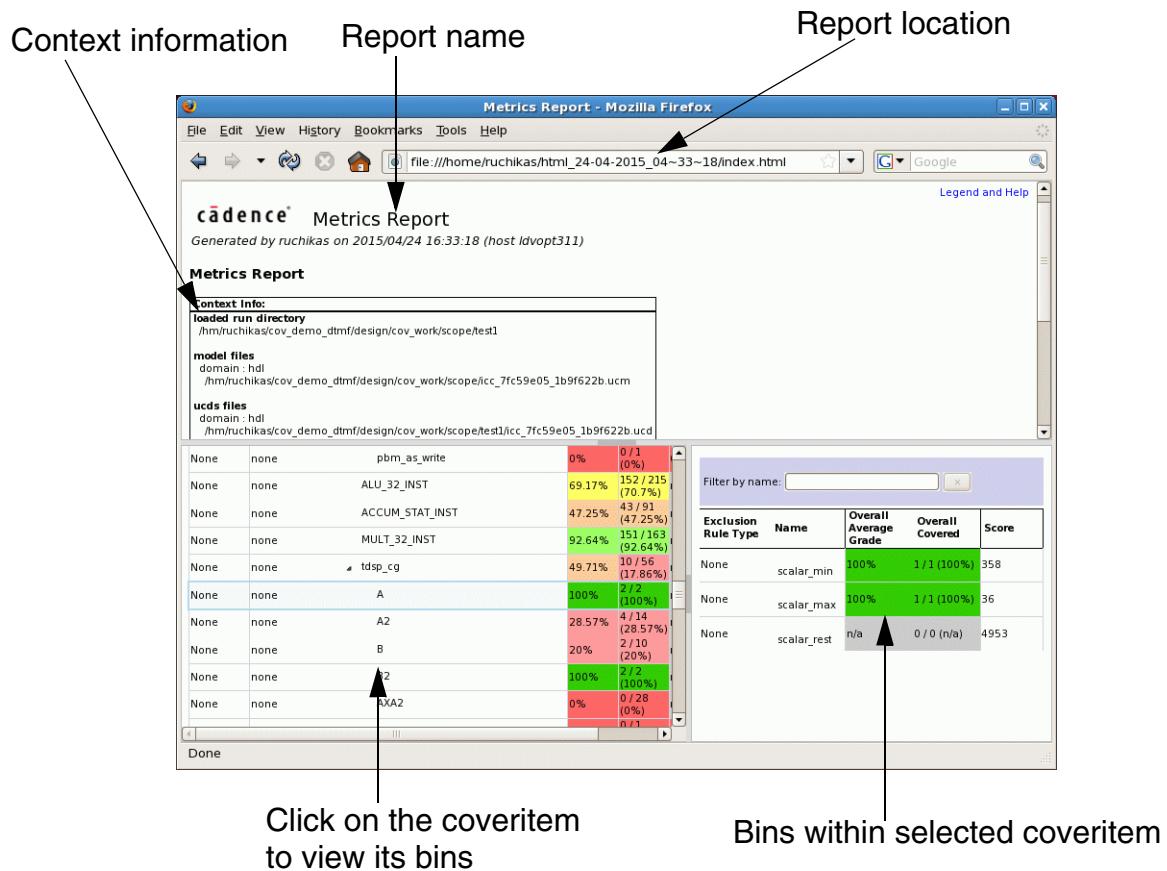
Metrics Report - Metrics Tree						
Current Node: /Verification Metrics/Instances/dtmf_recv.core/TDSP_CORE_INST						
Exclusion Rule Type	UNR	Name	Overall Average Grade	Overall Covered	Assertion Status Grade	
None	none	TDSP_CORE_INST	64.89%	2165 / 3620 (59.81%)	55%	
Sub-Nodes:						
Exclusion Rule Type	UNR	Name	Overall Average Grade	Overall Covered	Assertion Status Grade	
None	none	TDSP_CORE_GLUE_INST	75.77%	369 / 559 (66.01%)	n/a	
None	none	TDSP_CORE_MACH_INST	48.19%	45 / 98 (46.94%)	44%	Instances
None	none	DECODE_INST	78.9%	103 / 133 (77.44%)	n/a	
None	none	EXECUTE_INST	40.17%	467 / 1117 (41.81%)	n/a	
None	none	PROG_BUS_MACH_INST	62.56%	118 / 152 (77.63%)	66.67%	
None	none	DATA_BUS_MACH_INST	87.93%	132 / 160 (82.5%)	80%	
None	none	PORT_BUS_MACH_INST	17.91%	49 / 139 (35.25%)	0%	
None	none	ALU_32_INST	69.17%	152 / 215 (70.7%)	n/a	
None	none	ACCUM_STAT_INST	47.25%	43 / 91 (47.25%)	n/a	
None	none	MULT_32_INST	92.64%	151 / 163 (92.64%)	n/a	
None	none	tdsp_cg	49.71%	10 / 56 (17.88%)	n/a	
None	none	_sugar_assert_tdsp_phi_rel_6	0%	0 / 1 (0%)	0%	Covergroup
None	none	_sugar_assert_tdsp_phi_rel_5	0%	0 / 1 (0%)	0%	
None	none	_sugar_assert_tdsp_phi_rel_4	0%	0 / 1 (0%)	0%	Assertions

Notice that the assertions and covergroups are also listed in the hierarchy tree. These items are not listed if the *Show extend metrics tree* option is not enabled.

Example: Bin-Level Tabulated Report

Figure 7-10 on page 325 shows a bin-level tabulated report.

Figure 7-10 Metric Report (Tabulated)



The report shows information such as, who generated the report and when the report was generated. It also shows context information, such as coverage database location, model file (.ucm) and data file (.ucd) used to generate the report, the location and contents of the coverage configuration file, and the list of refinement files applied before generating the report.

The report shows two tables. The first table shows the *Metrics* hierarchy. You can view the bin-level information by selecting a cover item in the *Metrics* hierarchy. When you select a cover item, the bins associated with the coveritem are displayed in the table shown next to it.

The columns shown in the table are based on the columns available or view selected at the time of generating the report.

Both the tables in the report allows you to filter data, as required.

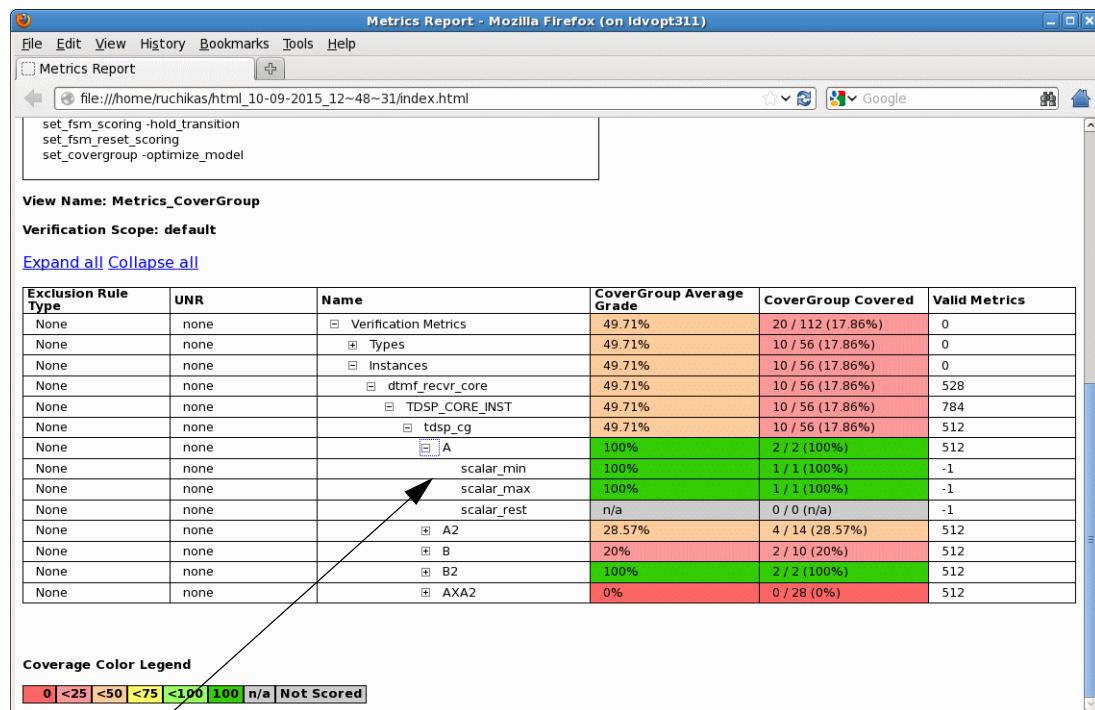
Note: Tabulated reports are supported in Firefox, Safari, and Internet Explorer. Chrome does not support loading data from local file system by default.

You can also generate this report from the CLI mode using [The report metrics Command](#) on page 391.

Example: Bin-Level Extended Report

[Figure 7-11](#) on page 326 shows an extended report.

Figure 7-11 Metric Report (Extended)



Bins Information shown in hierarchy tree

The report shows a single table. In this report, the *Metrics* tree and the bins information for each of the cover items is shown in the tree. You can expand or collapse the hierarchy tree using the links (Expand all and Collapse all) shown on the page.

Note: The expand/collapse capability in an extended report is supported only for trees with less than 4K items.

The columns shown in the table are based on the columns available or view selected at the time of generating the report.

Note: This report is a single HTML file and can be easily shared over mail in batch mode.

You can also generate this report from the CLI mode using [The report metrics Command](#) on page 391.

7.2 Generating Reports in Interactive Command-line Mode

The command-line interactive mode of IMC provides following two commands for generating reports:

- [The report Command](#) on page 327
- [The report_metrics Command](#) on page 391
- [The report_unique Command](#) on page 401

Note: The `report_metrics` command considers the configuration option *Show extend metrics tree* while generating the report. If the *Show extend metrics tree* option is enabled (using the *Configuration* dialog box in GUI or CLI `config` command), then the covergroups, cover items, FSMs, and assertions are also listed in the metrics hierarchy tree in the report. The *Show extend metrics tree* option is not considered for reports generated with the `report` command.

7.2.1 The report Command

The `report` command allows you to generate reports for data analysis in interactive command-line mode. By default, coverage reports generated with the `report` command are written to standard output unless redirected to a file. Using the `report` command, you can generate a:

- [List Report](#)
- [Summary Report](#)
- [Detailed Report](#)
- [HTML Reports](#)

Note: The `report` command is used to generate all the above type of reports. If the report type is not specified, then the `report` command generates a summary report.

7.2.1.1 List Report

Using the `report -list` command, you can generate an ASCII report showing a list of types or instances in the design. You can also list the instances or types for which specific coverage type is scored. The BNF of the `report -list` command is:

```
report -list
[-type | -inst]
[-metrics <metrics_type>]
[-out <name>]
[-append on|off]
<list>

metrics_type ::= [overall] [all] [code] [fsm] [functional] [block] [expression]
[toggle] [state] [transition] [arc] [covergroup] [assertion]
```

where

- `-inst | -type` specifies if report should be generated based on instance or type. If not specified, `-inst` is assumed.
- `-metrics <metrics_type>` specifies the type of coverage. With this option, the list report prints only the instances or types in which the specified `<metrics_type>` is scored. The `<metrics_type>` can be any, or a combination of the following:
 - `code` for listing types/instances in which block, expression, and toggle coverage is scored
 - `fsm` for listing types/instances in which state, transition, and arc coverage is scored
 - `functional` for listing types/instances in which both assertion and covergroup coverage is scored
 - `block` for listing types/instances in which block coverage is scored
 - `expression` for listing types/instances in which expression coverage is scored
 - `toggle` for listing types/instances in which toggle coverage is scored
 - `state` for listing types/instances in which state coverage is scored
 - `transition` for listing types/instances in which transition coverage is scored
 - `arc` for listing types/instances in which arc coverage is scored
 - `assertion` for listing types/instances in which PSL/SVA-based assertion coverage is scored
 - `covergroup` for listing types/instances in which SystemVerilog-based covergroup coverage is scored
 - `overall` for listing all of the types/instances

- `all` for listing all of the types/instances

You can specify more than one metrics type by separating the metric types with a colon (`:`). If not specified, all types/instances are listed.

- `-out <name>` specifies the name of the output file (`<name>`) where the report must be saved. By default, reports are written to standard output unless redirected to a file.
- `-append on|off` specifies if the output file must be appended or not. By default, the output file is overwritten. To append the output file instead of overwriting it, use the `-append on` option. This option is used only with the `-out` option.
- `<list>` specifies the names of types or instances for which the report will be generated. The default value is `* . . .`, which matches all the top-level instances and their children. It can include wildcard characters `*` and `?`. Use `. . .` to specify all descendants of an instance. Instance names can be specified as hierarchical names, for example, `top.inst1.inst2`. When specifying instance names that include characters like `[]` or `()`, use a backslash `\` to escape each such character. For example, to specify instance name `sys.ahb_evc.active_masters[1]monitor`, use:

```
report -list -inst sys.ahb_evc.active_masters\[1\]monitor
```

Note: The path can be prefixed with "*Verification\ Metrics*", "*Verification\ Metrics/Instances*", "*Verification\ Metrics/Types*", "*Instances*", or "*Types*". Prefix is not mandatory. However, if the *Instances* or *Types* prefix is specified, the performance of the command can be improved. For example, `report a.b.c` can be specified as:

```
report Verification\ Metrics/Instances/a.b.c
```

or

```
report Instances/a.b.c
```

Note: Wild cards (`*`, `?`, and `. . .`) are not supported in the prefix *Verification\ Metrics/Types* or *Verification\ Metrics/Instances*.

Note: The `report -list` command without any arguments resolves to `report -list -inst`.

Examples

To generate a list of instances in the loaded run, use:

```
report -list -inst
```

or

```
report -list
```

To list the types in the loaded run, use:

report -list -type

To list only the instances in which covergroups are scored, use:

```
report -list -metrics covergroup
```

To list only the instances in which FSMs are scored, use:

```
report -list -metrics fsm
```

[Figure 7-12](#) on page 330 shows the output of the above command.

Figure 7-12 Output of report -list -metrics fsm Command

metrics enabled	name
b e t f a c	dtmf_recv_core.RESULTS_CONV_INST
b e t f a c	dtmf_recv_core.TDSP_CORE_INST.TDSP_CORE_MACH_INST
b e t f a c	dtmf_recv_core.TDSP_CORE_INST.PROG_BUS_MACH_INST
b e t f a c	dtmf_recv_core.TDSP_CORE_INST.DATA_BUS_MACH_INST
b e t f a c	dtmf_recv_core.TDSP_CORE_INST.PORT_BUS_MACH_INST
b e t f a c	dtmf_recv_core.ARB_INST
b e t f a c	dtmf_recv_core.DMA_INST
b e t f a c	dtmf_recv_core.SPI_INST

Note: The command `report -list -metrics fsm` is the same as `report -list -metrics fsm -inst`.

In the above output, `b e t f a c` indicates the type of metrics enabled for scoring.

- b indicates block coverage
- e indicates expression coverage
- t indicates toggle coverage
- f indicates FSM coverage
- a indicates assertion coverage
- c indicates covergroup coverage

To generate a list of types that start with `p` in which FSMs are scored, use:

```
report -list -metrics fsm -type p*
```

The above command will report only the type names that start with `p` in which FSMs are scored.

To output the list of instances in which assertions are scored to a file `my_functional.txt`, use:

```
report -list -metrics assertion -out my_functional.txt
```

To append `my_functional.txt` file to include the list of instances in which covergroups are scored, use:

```
report -list -metrics covergroup -out my_functional.txt -append on
```

7.2.1.2 Summary Report

Summary report gives a quick snapshot view of the coverage numbers. The summary report shows coverage statistics as (Covered items / Total items /Marked items) for the specified metrics type. If there are no marked items, the report shows the coverage numbers as (Covered items / Total items).

To generate an ASCII summary report, use:

```
report [-summary]
[-type | -inst]
[-metrics <metrics_type>]
[-append on|off]
[-covered | -uncovered | -excludes | -unr | -all | -both]
[-cumulative on|off]
[-local on|off]
[-grading <average> | <covered> | <both>]
[-showempty on|off]
[-exclComments]
[-assertionStatus]
[format]
[-out <name>]
[-overwrite]
<list>
```

where

```
metrics_type ::= [overall] [all] [code] [fsm] [functional] [block] [expression]
[toggle] [state] [transition] [arc] [covergroup] [assertion]
format ::= [-text | -html]
```

In the above BNF,

- `-metrics <metrics_type>` specifies the type of coverage for which a quick snapshot view of the coverage numbers must be printed. The `<metrics_type>` can be any, or a combination of the following:
 - `overall` for printing coverage statistics of all metrics combined together
 - `all` for printing coverage statistics of all metrics (block, branch (if enabled at elaboration), statement (if enabled at elaboration), expression, toggle, FSM, assertion, covergroup) separately

- `code` for printing coverage statistics of block, expression, and toggle coverage combined together
- `fsm` for printing coverage statistics of state, transition, and arc coverage combined together
- `functional` for printing coverage statistics of assertion and covergroup coverage combined together
- `block` for printing coverage statistics of block coverage

Note: Use of `block` also prints a:

- Branch* column if branch coverage is enabled at elaboration.
- Statement* column if statement coverage is enabled at elaboration.

For more details on enabling branch coverage and statement coverage, see the *ICC User Guide*.

- `expression` for printing coverage statistics of expression coverage
- `toggle` for printing coverage statistics of toggle coverage
- `state` for printing coverage statistics of state coverage
- `transition` for printing coverage statistics of transition coverage
- `arc` for printing coverage statistics of arc coverage
- `assertion` for printing coverage statistics of PSL/SVA-based assertion coverage
- `covergroup` for printing coverage statistics of SystemVerilog-based covergroup coverage

If not specified, overall, code, fsm, and functional coverage is reported.

Note: You can specify more than one metric type by separating the metric types with a colon (:). For example, to report block and expression, use `block:expression`.

- `-all | -covered | -uncovered | -both | -excludes | -unr` specifies whether to list instances/types for all, covered, uncovered, both covered and uncovered, excluded, or unreachable items.

Note: If none of the options (`-all`, `-covered`, `-uncovered`, `-both`, `-unr`, or `-excludes`) is specified, `-uncovered` is assumed.

- `-cumulative on|off` enables or disables printing of cumulative grades in the report. By default, the summary report prints only the local grades and not the cumulative grades in the report. To report the cumulative grades along with the local grades, set the

-cumulative option to on. When you use the -cumulative on option, cumulative average grade as well as the cumulative covered grade is printed for the specified metrics type. This option is invalid and is ignored in the case of type-based reports.

For details on grade calculations, see [Coverage Grading in IMC](#).

- -grading <average> | <covered> | <both> lets you specify the cumulative coverage grade that must be printed in the summary report. By default, if cumulative grade printing is enabled (using the -cumulative on option), then both cumulative average grades as well as the cumulative covered grades are printed.
 - -grading average enables printing of only the cumulative average grade.
 - -grading covered enables printing of only the cumulative covered grade.
 - -grading both enables printing of both cumulative average grade as well as the cumulative covered grade. This is the default.

Note: In the absence of the -grading option, both is assumed. In addition, this option can be used only with the -cumulative option.

For details on grade calculations, see [Coverage Grading in IMC](#).

- -local on|off disables or enables reporting of local grades when cumulative grades are printed. By default, local grades are printed along with cumulative grades. To disable reporting of local grades, set the -local option to off. This option is used only with the -cumulative option.
- -showempty on|off enables or disables reporting of items with no self coverage data (0/0). By default, such items are not printed. To enable reporting of items with no self coverage, set the -showempty option to on.

Note: When the -showempty option is set to off, the entities with no self coverage are not reported unless they have a descendant that has coverage. Empty entities are reported if they have non-empty descendants.

- -exclComments enables reporting of exclusion comment in the summary report for the types or instances for which all coverage objects are excluded. By default, the exclusion comment is not printed in the report.

When you use this option in the report command, an additional column, Exclusion User, Reviewer, Comment is printed in the report, which shows the exclusion comment information as:

[<user>-<reviewer>] : <comment>

where

- <user> is the login ID of the user.

- ❑ <reviewer> is the name of the exclusion reviewer. The exclusion reviewer is specified using the `-reviewer` option of the `exclude` command. If no reviewer is specified at the time of exclusion, the default value `unknown` is considered as the reviewer. For more details on the `exclude` command, see [Refinement in Command-Line Interactive Mode](#) on page 188.
- ❑ <comment> is the exclusion comment. The exclusion comment is specified using the `-comment` option of the `exclude` command. For more details on the `exclude` command, see [Refinement in Command-Line Interactive Mode](#) on page 188.

For example, if the `exclude` command was specified as:

```
exclude -inst dtmf_recv_core.SPI_INST -comment "Excluding the complete instance" -reviewer Ben
```

and the `report` command was specified as:

```
report -summary -inst dtmf_recv_core.SPI_INST -exclComments
```

then an additional column `Exclusion User, Reviewer, Comment` with the following value will be printed in the report:

`[bensky-Ben] : Excluding the complete instance`

where `bensky` is the login ID, `Ben` is the exclusion reviewer, and `Excluding the complete instance` is the exclusion comment.

If the exclusion reviewer was not specified in the `exclude` command, then the `Exclusion User, Reviewer, Comment` column will show `[bensky-unknown] : Excluding the complete instance`.

Note: The `-exclComments` option enables printing of exclusion comments specified at the time of exclusion. In case exclusion comments are not specified at the time of exclusion and only the exclusion reviewer is specified, then the `Exclusion User, Reviewer, Comment` column will not show anything even if the `-exclComments` option is used.

- `-assertionStatus` enables reporting of assertion status information in the summary report. When you use this option, an additional column, `Assertion Status` is printed in the report, which shows the assertion status information for each instance or type as:

`Percentage% (Passed_assertions / Failed_assertions / Other_assertions)`

where

- ❑ `Percentage%` is the total percentage of passed assertions out of all assertions, which is calculated as `<Passed_assertions> / <Total_assertions> * 100`
- ❑ `Passed_assertions` is the total number of unexcluded assertions that finished without failure at least once, and never failed.

- ❑ `Failed_assertions` is the total number of unexcluded assertions that finished with failure at least once.
- ❑ `Other_assertions` is the total number of unexcluded assertions that never finished.

Note: In the case of cumulative coverage, column is named as `Assertion* Status`.

Note: The `-assertionStatus` option is applicable only for assertion coverage reports.

For an example of use of `-assertionStatus` option, see [Examples](#).

- `format` specifies the format of the report. It can be any of the following:
 - ❑ `-text` to generate report in a text format
 - ❑ `-html` to generate report in an HTML format.
- `-out <name>` specifies the location to which the report must be redirected.
 - ❑ If the `format` is specified as `-text`, the text reports are redirected to `<name>` instead of standard output. In this case, `<name>` is the name of the file to which the report is saved.
 - ❑ If the `format` is specified as `-html`, the HTML reports are redirected to `<name>` instead of default HTML output directory `html_<timestamp>`. In this case, `<name>` is the name of the directory that includes all the relevant files and folder required to show reports in the HTML format. For more details on HTML reports, see [HTML Reports](#) on page 370.
- `-overwrite` enables overwriting of the existing HTML output directory. By default, HTML output directory is not overwritten and an error is reported.

Note: The `-overwrite` option is ignored in the case of text reports.

See [List Report](#) for details on syntax description of other options.

Examples

To generate an instance-based block summary report with cumulative and local grades, use:

```
report -summary -metrics block -cumulative on
```

[Figure 7-13](#) on page 336 shows the output of the above command.

Figure 7-13 Instance-Based Summary Report

imc> report -summary -metrics block -cumulative on

Legend: Metric* means cumulative e.g. Block* means Cumulative Block Coverage

name	Block* Average	Block* Covered	Block
dtmf_recv_core	74.92%	49.75% (490/985/16)	n/a
--TEST_CONTROL_INST	50.00%	50.00% (5/10)	50.00% (5/10)
--ROM_512x16_INST	n/a	n/a	n/a (0/0/5)
--RAM_128x16_TEST_INST	75.00%	85.71% (6/7/3)	50.00% (1/2)
--RAM_128x16_INST	100.00%	100.00% (5/5/3)	100.00% (5/5/3)
--RAM_256x16_TEST_INST	75.00%	80.00% (4/5/3)	50.00% (1/2)
--RAM_256x16_INST	100.00%	100.00% (3/3/3)	100.00% (3/3/3)
--TDSP_DS_CS_INST	64.29%	64.29% (9/14)	64.29% (9/14)
--DATA_SAMPLE_MUX_INST	100.00%	100.00% (9/9)	100.00% (9/9)
--DIGIT_REG_INST	66.67%	66.67% (2/3)	66.67% (2/3)
--RESULTS_CONV_INST	10.00%	10.00% (10/100)	10.00% (10/100)

In the above report, following information is printed:

- The `name` column indicates the name of the instance or type. In the absence of `-type` or `-inst` option on the command line, `-inst` is assumed and an instance-based report is generated.
- The column with the metrics name, for example, `Block` shows the block local grade.
- The column name with a * for example, `Block*` indicates cumulative grades. This information is printed due to the use of `-cumulative` option on the command line. `Block * Average` shows the *Block Cumulative Average* grade and `Block * Covered` shows the *Block Cumulative Covered* grade. The absolute coverage numbers are also shown in the *Block Cumulative Covered* grade as (Covered items / Total items /Marked items). If there are no marked items, the report shows the absolute numbers as (Covered items / Total items).

Note: If a few items are marked unreachable by Incisive Enterprise Verifier (IEV), then the absolute coverage numbers are shown as:

(Covered items / Total items (UNR items) / Marked items)

where `UNR items` is the number of unreachable items.

For example, [Figure 7-14](#) on page 337 shows a block summary report with UNR items.

Figure 7-14 Block Summary Report with UNR Items

name	Block* Average	Block* Covered	Block
top	88%	93% (14/15(1)/1)	100% (11/11)
--c1	75%	75% (3/4(1)/1)	75% (3/4(1)/1)

Value in brackets () indicates number of UNR items

Note: UNR entities are counted in the overall count (unlike excluded entities).

For details on the UNR flow and how items are marked UNR by IEV, see the *IEV User Guide*.

To print only cumulative grades in the above report, use:

```
report -summary -metrics block -cumulative on -local off
```

Figure 7-15 on page 337 shows the output of the above command.

Figure 7-15 Summary Report with Cumulative Grades

name	Block* Average	Block Coverage
dtmf_recv_core	74.92%	49.75% (490/985/16)
--TEST_CONTROL_INST	50.00%	50.00% (5/10)
--ROM_512x16_INST	n/a	n/a
--RAM_128x16_TEST_INST	75.00%	85.71% (6/7/3)
--RAM_128x16_INST	100.00%	100.00% (5/5/3)
--RAM_256x16_TEST_INST	75.00%	80.00% (4/5/3)
--RAM_256x16_INST	100.00%	100.00% (3/3/3)
--TDSP_DS_CS_INST	64.29%	64.29% (9/14)
--DATA_SAMPLE_MUX_INST	100.00%	100.00% (9/9)
--DIGIT_REG_INST	66.67%	66.67% (2/3)
--RESULTS_CONV_INST	10.00%	10.00% (10/100)

In the above report, local grades are not printed due to use of the `-local off` option.

For details on how grades are calculated, see [Coverage Grading in IMC](#) on page 407.

To print only block cumulative average grade in the above report, use:

```
report -summary -metrics block -cumulative on -local off -grading average
```

Figure 7-16 on page 338 shows the output of the above command.

Figure 7-16 Summary Report with only Cumulative Average Grades

name	Block* Average
dtmf_recv_core	74.92%
- TEST_CONTROL_INST	50.00%
- ROM_512x16_INST	n/a
- RAM_128x16_TEST_INST	75.00%
- - RAM_128x16_INST	100.00%
- RAM_256x16_TEST_INST	75.00%
- - RAM_256x16_INST	100.00%
- TDSP_DS_CS_INST	64.29%
- DATA_SAMPLE_MUX_INST	100.00%
- DIGIT_REG_INST	66.67%
- RESULTS_CONV_INST	10.00%

In the above report, only the cumulative average grade is printed due to the use of the -grading average option.

To generate an type-based functional coverage summary report, use:

```
report -summary -metrics functional -type
```

Figure 7-17 on page 338 shows the output of the above command.

Figure 7-17 Type-Based Summary Report

Module names	Local Functional Average Grade		Local Functional Covered Grade
	Functional	Average	
tdsp_core	39.14%	19.05% (12/63)	
tdsp_core_mach	100.00%	100.00% (3/3)	
prog_bus_mach	66.67%	66.67% (2/3)	
data_bus_mach	100.00%	100.00% (5/5)	
port_bus_mach	0.00%	0.00% (0/2)	
arb	72.73%	72.73% (8/11)	
dma	42.86%	42.86% (3/7)	
spi	66.67%	66.67% (6/9)	

The above report shows modules in which functional coverage is found.

To generate assertion coverage summary report with assertion status information, use:

```
report -summary -metrics assertion -assertionStatus
```

Figure 7-18 on page 339 shows the output of the above command.

Figure 7-18 Assertion Coverage Summary Report with Assertion Status Information

Assertion covered		Assertion status information	
imc> report -summary -metrics assertion -assertionStatus name		Assertion	Assertion Status

dtnf_recv_core	n/a	n/a	
--TDSP_CORE_INST	28.57% (2/7)	29% (2/0/5)	
--TDSP_CORE_MACH_INST	100.00% (3/3)	100% (3/0/0)	
--PROG_BUS_MACH_INST	66.67% (2/3)	67% (2/0/1)	
--DATA_BUS_MACH_INST	100.00% (5/5)	80% (4/1/0)	
--PORT_BUS_MACH_INST	0.00% (0/2)	0% (0/0/2)	
--ARB_INST	72.73% (8/11)	64% (7/1/3)	
--DMA_INST	42.86% (3/7)	43% (3/3/1)	
--SPI_INST	66.67% (6/9)	56% (5/3/1)	

(Passed / Failed / Other)

In the above report, assertion status is also printed along with the assertion covered grade. In the Assertion Status column, for example, notice the value corresponding to instance ARB_INST. The value is printed as 64% (7/1/3). This indicates that:

- 64% is the total percentage of passed assertions out of all assertions, which is calculated as $(7 / 11) * 100$
- 7 assertions were categorized as Passed (finished without failure at least once, and never failed).
- 1 assertion was categorized as Failed (finished with failure at least once).
- 3 assertions were categorized as Other (never finished).

If cumulative coverage is also turned ON (using the `-cumulative on` option), then column named Assertion* Status is also printed in the report, as shown in [Figure 7-19](#) on page 339.

Figure 7-19 Assertion Coverage Summary Report with Cumulative Assertion Status

		Assertion Status (cumulative)		Assertion Status (local)	
imc> report -summary -metrics assertion -assertionStatus -cumulative on					
Legend: Metric* means cumulative e.g. Block* means Cumulative Block Coverage					
name		Assertion* Average	Assertion* Covered	Assertion* Status	Assertion

dtnf_recv_core	60.32%	61.70% (29/47)	55% (26/8/13)	n/a	n/a
--TDSP_CORE_INST	59.05%	60.00% (12/20)	55% (11/1/8)	28.57% (2/7)	29% (2/0/5)
--TDSP_CORE_MACH_INST	100.00%	100.00% (3/3)	100% (3/0/0)	100.00% (3/3)	100% (3/0/0)
--PROG_BUS_MACH_INST	66.67%	66.67% (2/3)	67% (2/0/1)	66.67% (2/3)	67% (2/0/1)
--DATA_BUS_MACH_INST	100.00%	100.00% (5/5)	80% (4/1/0)	100.00% (5/5)	80% (4/1/0)
--PORT_BUS_MACH_INST	0.00%	0.00% (0/2)	0% (0/0/2)	0.00% (0/2)	0% (0/0/2)
--ARB_INST	72.73%	72.73% (8/11)	64% (7/1/3)	72.73% (8/11)	64% (7/1/3)
--DMA_INST	42.86%	42.86% (3/7)	43% (3/3/1)	42.86% (3/7)	43% (3/3/1)
--SPI_INST	66.67%	66.67% (6/9)	56% (5/3/1)	66.67% (6/9)	56% (5/3/1)

In this report, cumulative assertion status is also printed. For details on how cumulative grades are calculated, see [Coverage Grading in IMC](#) on page 407.

Note: Similarly, you can generate summary reports for other metric types.

If the design includes generate blocks, then the generate blocks are listed separately, as shown in [Figure 7-20](#) on page 340.

Figure 7-20 Type-Based Summary Report

name	Block	Branch	Expression
worklib.Exprl(Arch_exprl)	80% (4/5)	50% (1/2)	50% (7/14)
WORKLIB_Exprl(Arch_exprl)_GK_GK0	80% (4/5)	50% (1/2)	14% (1/7)
BP	25% (3/12)	25% (3/12)	37% (23/62)

Generate block

Note: If the generate block is not named explicitly, the tool automatically generates a name for the generate block, and displays it in the report. In addition, in the report, the grade is showing no decimal places because the value of `float_precision` configurable item is set to 0. By default, the value is set to 2 and you would see 2 decimal places in the attributes that show values as decimals. For more details, see [Setting Values for Configurable Items](#) on page 254.

7.2.1.3 Detailed Report

The detailed report lists details for all or selected types/design entity or instances in the design. To generate an ASCII detailed report, use:

```
report -detail
[-type | -inst]
[format]
[-metrics <metrics_type>]
[-out <name>]
[-overwrite]
[-covered | -uncovered | -excludes | -unr | -all | -both]
[-kind <abstract>]
[-append on|off]
[-showempty on|off]
[-source on|off]
[-aspect sim | formal | both]
[-assertionStatus]
[-allAssertionCounters]
[-exclComments]
<list>
```

where

```
metrics_type ::= [overall] [all] [code] [fsm] [functional] [block] [expression]
[toggle] [state] [transition] [arc] [covergroup] [assertion]
```

```
format ::= [-text | -html]
```

In the above BNF:

- `-all | -covered | -uncovered | -both | -excludes | -unr` specifies whether to generate reports for all, covered, uncovered, both covered and uncovered, excluded, or unreachable items.
- Note:** If none of the options (`-all`, `-covered`, `-uncovered`, `-both`, `-unr`, or `-excludes`) is specified, `-uncovered` is assumed. However, in the case of covergroup detailed report in an HTML format, if none of these options are specified, `-both` is assumed.
- `-metrics <metrics_type>` specifies the type of coverage for which a detailed report must be printed. The `<metrics_type>` can be any, or a combination of the following:
 - `code` for printing a detailed block, expression, and toggle coverage report
 - `fsm` for printing a detailed state, transition, and arc coverage report
 - `functional` for printing a detailed assertion and covergroup coverage report
 - `block` for printing a detailed block coverage report
 - `expression` for printing a detailed expression coverage report
 - `toggle` for printing a detailed toggle coverage report
 - `state` for printing a detailed state coverage report
 - `transition` for printing a detailed transition coverage report
 - `arc` for printing a detailed arc coverage report
 - `assertion` for printing a detailed assertion coverage report
 - `covergroup` for printing a detailed covergroup coverage report
 - `overall` for printing a detailed report for all metrics
 - `all` for printing a detailed report for all metrics
- **Note:** You can specify more than one metrics type by separating the metric types with a colon (:). If not specified, a detailed report for all metrics is generated.
- `-kind <abstract>` disables printing of bin information from the ASCII detailed covergroup reports. By default, bin information is printed in the detailed ASCII report.

- `-source off | on` enables or disables reporting of the source text and line numbers from the source code. By default, this information is reported. To disable printing of source text and line numbers, set the `-source` option to `off`.

Note: Currently, the batch mode of IMC cannot read the compressed files, and therefore, the *Source Code* column of the report of entities from the compressed files will be blank (even if the `-source` option is turned `on`).

- `-aspect sim | formal | both` specifies the assertion properties to be shown in the report.
 - `sim` shows only simulation assertion properties in the report.
 - `formal` shows only formal assertion properties in the report.
 - `both` shows both simulation and formal assertion properties in the report.

Note: By default, the value of the `-aspect` option is set to `sim`.

- `-assertionStatus` enables reporting of assertion status information in the detailed report. When you use this option, additional column(s) is shown in the report depending on the aspect specified with the `-aspect` option. The report shows following additional columns:
 - Status** (if `-aspect` is `sim`). The Status can be any of the following:
 - `Passed` if the assertion finished without failure at least once, and never failed.
 - `Failed` if the assertion finished with failure at least once.
 - `Other` if the assertion never finished.
 - Formal Status** (if `-aspect` is `formal`). The Formal Status can be any of the following:
 - `Proved` if the assertion finished without failure at least once, and never failed.
 - `Failed` if the assertion finished with failure at least once.
 - `Other` if the assertion never finished.

Note: If `-aspect` is specified as `both`, then both *Status* and *Formal Status* attributes are printed in the report.

Note: If `-allAssertionCounters` option is also used with the `-assertionStatus` option, then the value `Other` in the *Status* field is further classified as any of the following:

- `Vacuous` if the assertion was vacuous pass at least once but never finished.

- Disabled if the assertion transitioned to the disabled state at least once but never finished.
- Attempted if the assertion was attempted at least once but never finished.
- Not Attempted if the assertion was not attempted at all (that is all the counters are equal to zero).

Note: The `-assertionStatus` option is applicable only for assertion coverage reports.

- `-allAssertionCounters` enables reporting of additional columns in the detailed report.
 - In case `-aspect` is `sim`, then Vacuous, Attempt, and Disabled counters are also reported in addition to the default Finished and Failed counters.
 - In case `-aspect` is `formal`, then Formal Proved, Formal Passed, and Formal CEX counters are also reported in addition to the default Formal Finished and Formal Failed counters.

Note: In case `-aspect` is specified as `both`, then counters reported with both (`sim` and `formal`) are printed in the report.

Note: The `-allAssertionCounters` option is applicable only if the `-abvrecordvacuous` switch is used at the time of simulation run. In case the `-abvrecordvacuous` switch is not used at simulation, then the Vacuous, Attempt, and Disabled columns will show n/a. For more details on the `-abvrecordvacuous` switch, see the *Verilog Compilation Command-Line Options* user guide.

For examples on Assertion coverage report, see [Examples: Assertion Coverage](#) on page 364.

- `-exclComments` enables reporting of exclusion comment in the detailed report. By default, this information is not printed in the report. When you use this option in the `report` command, an additional column, Exclusion User, Reviewer, Comment is printed in the detailed report, which shows the exclusion comment information as:

`[<user>-<reviewer>] : <comment>`

where

- `<user>` is the login ID of the user.
- `<reviewer>` is the name of the exclusion reviewer. The exclusion reviewer is specified using the `-reviewer` option of the `exclude` command. If no reviewer is specified at the time of exclusion, the default value `unknown` is considered as the reviewer. For more details on the `exclude` command, see [Refinement in Command-Line Interactive Mode](#) on page 188.

- ❑ <comment> is the exclusion comment. The exclusion comment is specified using the `-comment` option of the `exclude` command. For more details on the `exclude` command, see [Refinement in Command-Line Interactive Mode](#) on page 188.

For example, if the `exclude` command was specified as:

```
exclude -inst dtmf_recv_core.SPI_INST -block 6 -comment "This block cannot be
covered" -reviewer Ben
```

and the `report` command was specified as:

```
report -detail -metrics block -all -inst dtmf_recv_core.SPI_INST
-exclComments
```

then an additional column `Exclusion User,Reviewer,Comment` will be printed in the report, as shown in [Figure 7-21](#) on page 344.

Figure 7-21 Sample Report

Count	Block	Line	Kind	Origin	Source Code	Exclusion User,Reviewer,Comment
IGN	1	92	code block	91	if (bit_cnt_reset)	
IGN	2	93	true part of	*	92 bit_cnt <= 0 :	
IGN	3	94	false part of	*	92 else if (not_full)	
IGN	4	95	true part of	*	94 bit_cnt <= bit_cnt + 1 :	
IGN	5	94	implicit else	*	94 else if (not_full)	
EXCL	6	101	code block	100	if (not_full)	[ruchikas-Ben]: This block cannot be covered

In the `Exclusion User,Reviewer,Comment` column, notice the value shown for block 6. In this case, ruchikas is the login ID, Ben is the exclusion reviewer, and This block cannot be covered is the exclusion comment.

If the exclusion reviewer was not specified in the `exclude` command, then the `Exclusion User,Reviewer,Comment` column would have shown [ruchikas-unknown]: This block cannot be covered.

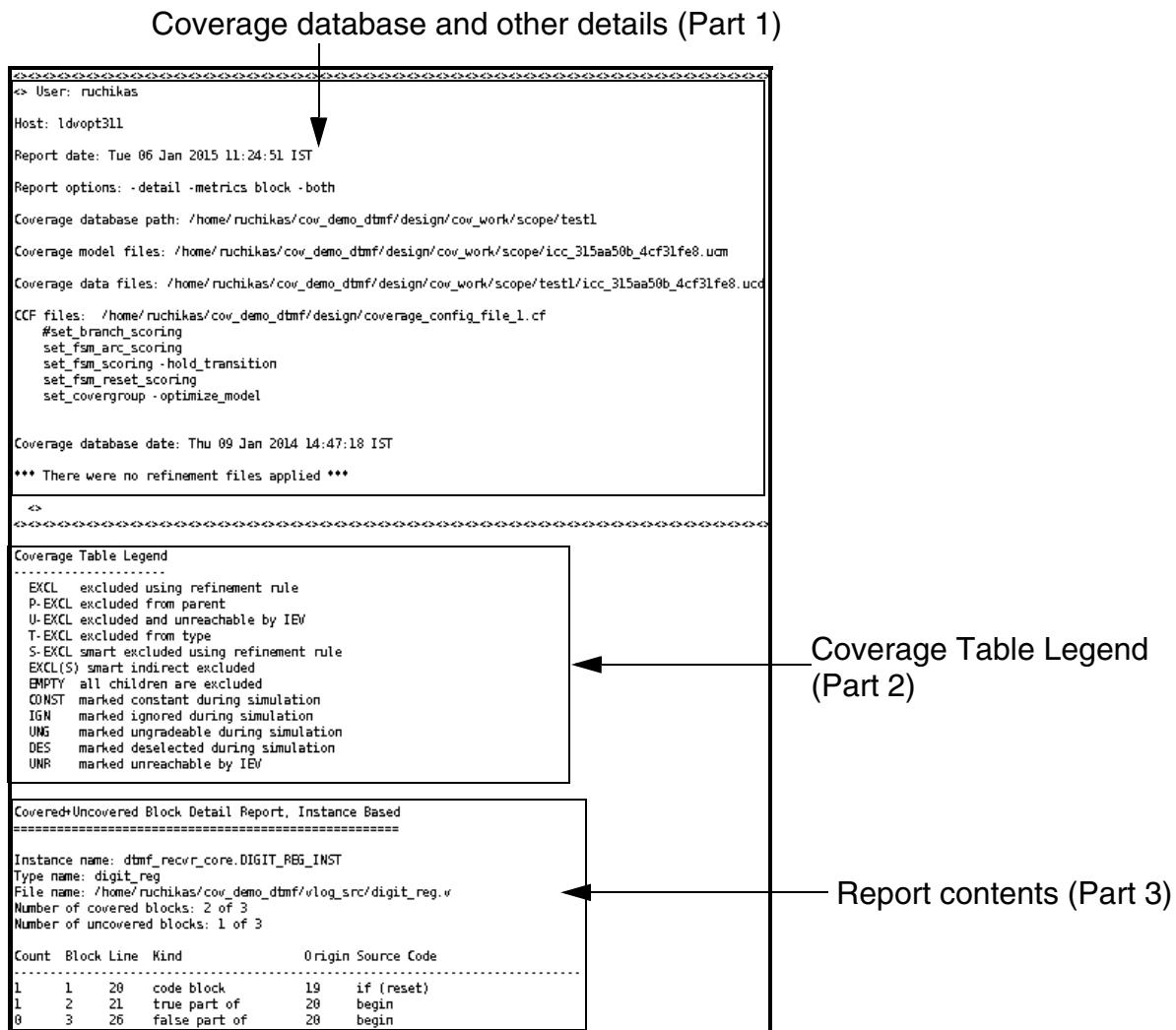
Note: The `-exclComments` option enables printing of exclusion comments specified at the time of exclusion. In case exclusion comments are not specified at the time of exclusion and only the exclusion reviewer is specified, then the `Exclusion User,Reviewer,Comment` column will not show anything even if the `-exclComments` option is used.

See [Summary Report](#) for details on syntax description of other options.

Sample Detailed Report

[Figure 7-22](#) on page 345 shows a sample detailed report.

Figure 7-22 Sample Detailed Report



A detailed report is divided in three parts:

- Part 1 — Shows the information, such as who generated the report, when the report was generated, the options used to generate the report, coverage database location, model file (.ucm) and data file (.ucd) used to generate the report, the location and contents of the coverage configuration file, time when the coverage database was generated, and the list of refinement files applied before generating the report.
- Part 2 — Shows the following coverage table legend information:
 - EXCL—Represents items that are excluded using a refinement rule
 - P-EXCL—Represents items that are excluded from the parent

- ❑ U-EXCL—Represents excluded items that are marked as unreachable by IEV
- ❑ T-EXCL—Represents items that are excluded from the type
- ❑ S-EXCL—Represents items that are smart excluded using refinement rule
- ❑ EXCL (S)—Represents items that are smart indirect excluded
- ❑ EMPTY—Indicates that all children are excluded
- ❑ CONST—Represents items marked constant during simulation
- ❑ IGN—Represents items marked ignored during simulation
- ❑ UNG—Represents the items that are marked ungradable during simulation
- ❑ DES—Represents the items that are marked deselected during simulation
- ❑ UNR—Represents the items that are marked unreachable by IEV
- ❑ UNG-Expr—Represents the expression marked ungradeable during simulation
- ❑ UNG-Toggle—Represents the toggle marked ungradeable during simulation
- Part 3 — Shows the report contents, such as coverage counts, line number, instance name, type name, and so on, depending on the metrics for which the report is generated.

Note: Part 1 and Part 2 are common and are printed in each report. Part 3 of the report is based on the metrics type selected and other options specified while generating the report. The following topics cover the sample reports of each of the metrics type. The reports shown in examples below show only part 3 of the report.

Example: Block Coverage

To print a block coverage report for the instance `DIGIT_REG_INST`, use:

```
report -detail -metrics block -both dtmf_recv_core.DIGIT_REG_INST
```

Figure 7-23 on page 347 shows the output of the above command.

Figure 7-23 Block Coverage Detailed Report

Number of times a block was exercised

Design file

Covered+Uncovered Block Detail Report, Instance Based

=====
 Instance name: dtmf_recv_core.DIGIT_REG_INST
 Type name: digit_reg
 File name: /home/ruchikas/cov_demo_dtmf/vlog_src/digit_reg.v
 Number of covered blocks: 2 of 3
 Number of uncovered blocks: 1 of 3
 Number of covered branches: 1 of 2
 Number of uncovered branches: 1 of 2

Count Block # Stmt Line Kind Origin Source Code

Block ID	Block #	Stmt	Line	Kind	Origin	Source Code
1	1	1	20	code block	19	if (reset)
1	2	2	21	true part of	*	begin
0	3	2	26	false part of	*	begin

(*) indicates a branch

Statements in each block

Type of block

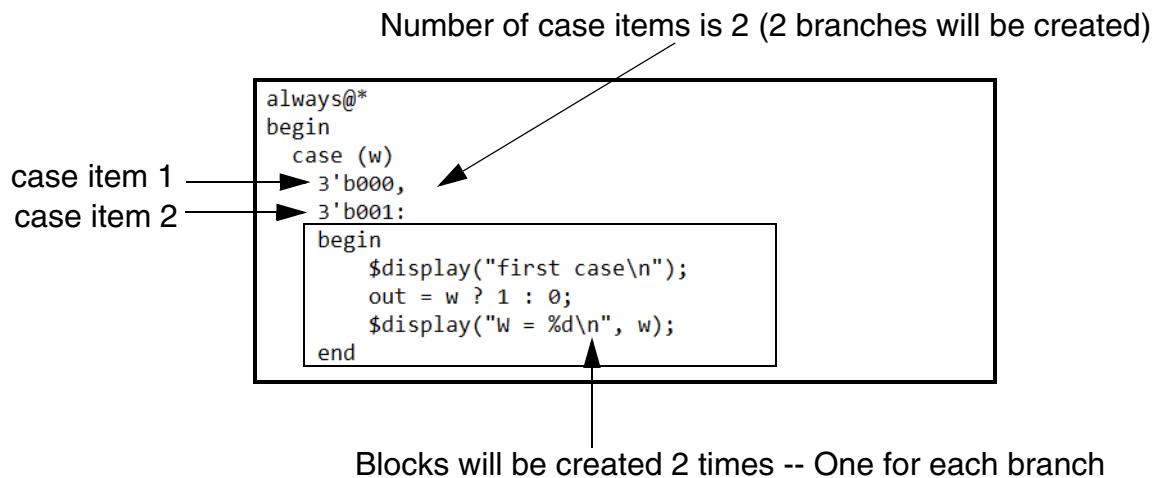
Source line number

The above report is generated after enabling branch and statement coverage.

Note: In a block detailed report, the branches are identified by descriptions next to them as ternary, true part of, false part of, a case item of, or implicit else. The above report has branches on Line 21 and Line 26 (which are also scored as blocks).

Note: Each case item is treated as a separate branch (if branch scoring is enabled at elaboration). For example, consider the code shown in [Figure 7-24 on page 348](#).

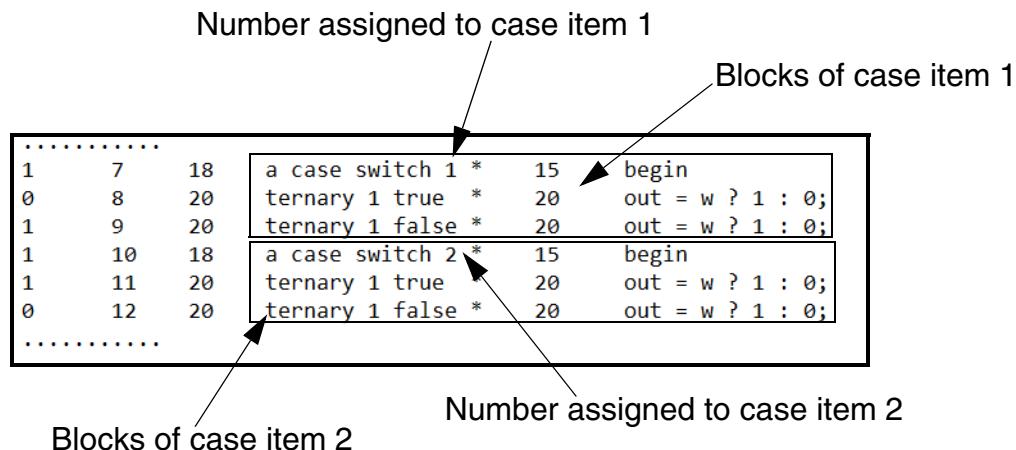
Figure 7-24 Sample Code



In this case, the two case items -- 3 'b000 (case item 1) and 3 'b001 (case item 2), execute the same set of RTL statements. As a result, blocks will be created two times, one for each branch.

The tool assigns a number to each case item based on the RTL, that is -- whichever case item is written first in the RTL is numbered first. The number assigned to the case item is written at the start of each case block in the report, as shown in [Figure 7-25 on page 348](#).

Figure 7-25 Report with Case Items



Notice that even though there are multiple blocks on the same line but the case items are different for each block. As a result, there is no logical repetition of the code block.

If the design includes generate blocks, then the name of the generate block is appended to the name of the type, as shown in [Figure 7-26](#) on page 349.

Figure 7-26 Type-Based Detailed Report

Name of the generate block appended to the type name

↓

Type name: WORKLIB_Expr1(Arch_expr1). EX_GKO						
File name: /vobs/mac_test/test/mac_batch_test/test_designs/vhdl/block_expr_coverage/fine_grain_merging/generate/test2/stim_drive1.vhd						
Number of covered blocks: 4 of 5						
Number of uncovered blocks: 1 of 5						
Number of covered branches: 1 of 2						
Number of uncovered branches: 1 of 2						
Count	Block	Line	Kind	Origin	Source	Code
16	1	67	code block	65	expr(output, input1, input2, input3);	
1	2	75	code block	73	IF (input1 = '1') THEN	
1	3	75	implicit else *	75	IF (input1 = '1') THEN	
0	4	76	true part of *	75	ou(11) <= input2 and input3;	
1	5	78	code block	73	WAIT UNTIL (INPUT2 = INPUT3);	

Example: Blocks Marked Unreachable by IEV

If a block is marked unreachable by IEV, then the *Count* column shows UNR, as shown in [Figure 7-27](#) on page 349.

Figure 7-27 Instance-Based Detailed Report (with items marked UNR)

Number of blocks marked unreachable →

Instance name: top.cl						
Type name: can_dataer						
File name: /home/ruchikas/hdlsynth/testcases_IMC/unr_flow_example/unr_flow_example/test.v						
Number of covered blocks: 4 of 5						
Number of uncovered blocks: 1 of 5						
Number of excluded blocks: 0						
Number of unreachable blocks: 1						
Count	Block	Line	Kind	Origin	Source	Code
1	1	9	code block	8	if (reset)	
1	2	10	true part of	9	left <= 0;	
1	3	11	false part of	9	else if (load && dispense)	
UNR	4	12	true part of	11	left <= data;	
1	5	11	implicit else	11	else if (load && dispense)	

↑ Block marked unreachable by IEV

In the above report, the *Count* column shows UNR for block 4 because it was determined as unreachable by IEV. In addition, the report header also shows the number of unreachable blocks.

Note: The UNR markers are also shown in the HTML-based report.

For details on the UNR flow and how items are marked UNR by IEV, see the *IEV User Guide*.

Example: Expression Coverage

To print a detail report, displaying expression coverage data for the instance

dtmf_recv_core.TDSP_CORE_INST.DECODE_INST, use:

```
report -detail -metrics expression -both
dtmf_recv_core.TDSP_CORE_INST.DECODE_INST
```

[Figure 7-28](#) on page 350 shows the output of the above command.

Figure 7-28 Expression Coverage Detailed Report

Coverage information of evaluated expressions																								
Covered+Uncovered Expression Detail Report, Instance Based =====																								
Expression coverage Table Legend -----																								
- don't care e event for event-or expressions 0 (odd), E (even), B (both), X (not scored), I (marked ignore) for parity trees Y (covered), N (not covered), C (constant), P (one or more inputs for this bit have been padded) for vector scoring, d==, b== shows which bit differs in vector scoring rval Resulting value of the expression for coverage purposes given the input values <-n-> Shows the n-th term composition																								
Instance name: dtmf_recv_core.TDSP_CORE_INST.DECODE_INST Type name: decode_i File name: /home/ruchikas/cov_demo_dtmf/vlog_src/decode_i.v Number of covered expressions: 5 of 6 Number of uncovered expressions: 1 of 6																								
<table border="1"> <thead> <tr> <th>index</th><th>grade</th><th>line</th><th>expression</th><th></th></tr> </thead> <tbody> <tr> <td>1.1</td><td>100.00% (3/3)</td><td>127</td><td>phi_6 && (! skip_one)</td><td></td></tr> <tr> <td>2.1</td><td>66.67% (2/3)</td><td>131</td><td>(! two_cycle) && (! decode_skip_one)</td><td></td></tr> </tbody> </table>					index	grade	line	expression		1.1	100.00% (3/3)	127	phi_6 && (! skip_one)		2.1	66.67% (2/3)	131	(! two_cycle) && (! decode_skip_one)						
index	grade	line	expression																					
1.1	100.00% (3/3)	127	phi_6 && (! skip_one)																					
2.1	66.67% (2/3)	131	(! two_cycle) && (! decode_skip_one)																					
index: 1.1 grade: 100.00% (3/3) line: 127 source: if (phi_6 && ! skip_one) phi_6 && (! skip_one) <-1-> <-2->																								
<table border="1"> <thead> <tr> <th>index</th><th>hit</th><th>rval</th><th><1></th><th><2></th></tr> </thead> <tbody> <tr> <td>1.1.1</td><td>1</td><td>1</td><td>1</td><td>0</td></tr> <tr> <td>1.1.2</td><td>1</td><td>0</td><td>-</td><td>1</td></tr> <tr> <td>1.1.3</td><td>1</td><td>0</td><td>0</td><td>-</td></tr> </tbody> </table>					index	hit	rval	<1>	<2>	1.1.1	1	1	1	0	1.1.2	1	0	-	1	1.1.3	1	0	0	-
index	hit	rval	<1>	<2>																				
1.1.1	1	1	1	0																				
1.1.2	1	0	-	1																				
1.1.3	1	0	0	-																				
# times term evaluated hit Don't care bits can take any value (0 or 1)																								

In the expression coverage detailed report following expression coverage table legend information is printed:

- - — Indicates don't care bits
- e—Indicates event for event-or expressions
- 0—Represents odd bit in a parity tree

- E—Represents even bit in a parity tree
- B—Represents both even and odd bits in a parity tree
- X—Indicates that the bit is not scored in a parity tree
- I—Indicates that the bit is marked ignored in a parity tree
- Y—Indicates that the bit is covered
- N—Indicates that the bit is not covered
- C—Indicates that the bit is constant
- P—Indicates that one or more bits of this input have been padded for vector scoring
- d==, b==—Indicates which bit differs in vector scoring
- rval—Displays the resulting value of expression
- <-n->—Shows the n^{th} term composition

Below the coverage legend, the expression coverage report shows a snapshot view of all of the evaluated expressions. Consider a sample snapshot view in [Figure 7-29](#) on page 351.

Figure 7-29 Snapshot View of Evaluated Expressions

index	grade	line	expression
1.1	33% (1/3)	231	(bist_cntrl1_reg[0] == 1) && (bist_old_req == 0)
1.2	50% (1/2)	231	(bist_cntrl1_reg[0] == 1)
1.5	0% (0/2)	231	(bist_old_req == 0)

This portion of the report prints the index, grade, line number, expression for the expressions evaluated during simulation run. The `index` column prints the index as:

`<top-expr-id>.<sub-expr-id>`

where:

- `<top-expr-id>` is the identifier of the top-level expression.
- `<sub-expr-id>` is the identifier of the sub-level expression.

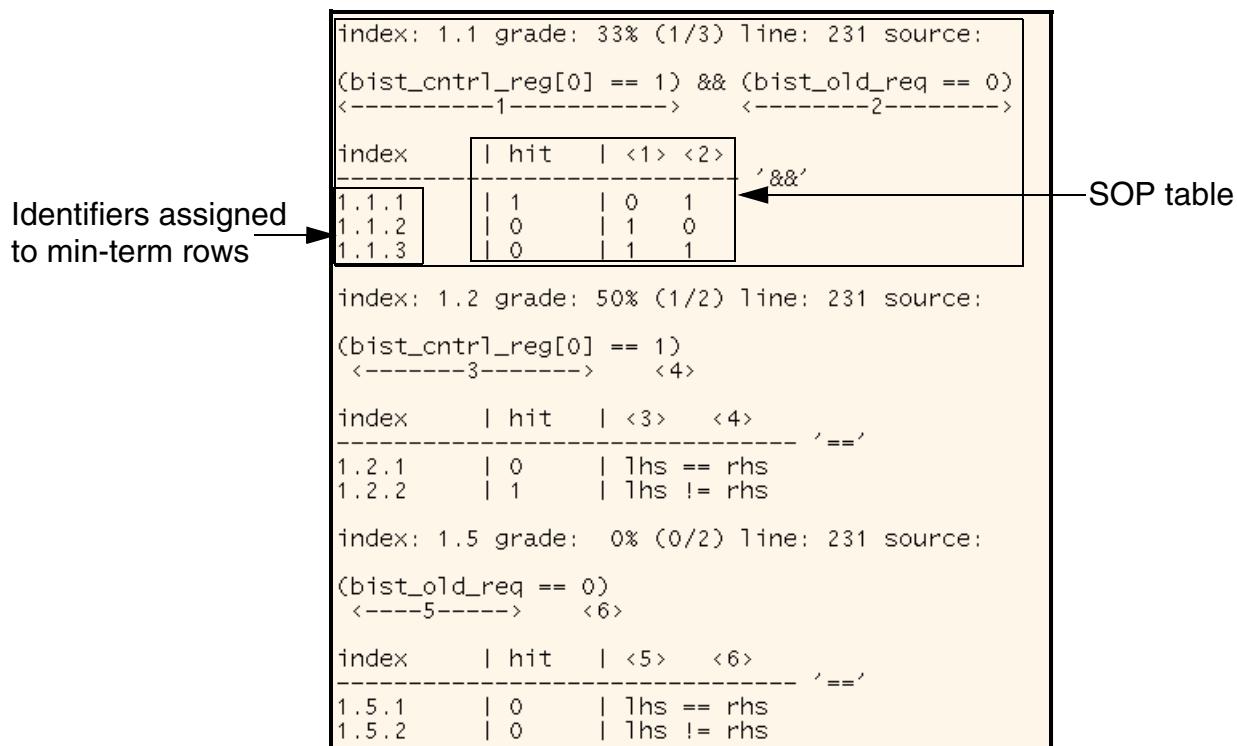
Note: The expression coverage report does not print information of expressions for which an SOP table is not generated. For example, in the above report, notice that index 1.5 is printed immediately after 1.2. This is because there is no SOP table for 1.3 and 1.4.

Note: In the report, the grade is showing no decimal places because the value of `float_precision` configurable item is set to 0. By default, the value is set to 2 and you

would see 2 decimal places in the attributes that show values as decimals. For more details, see [Setting Values for Configurable Items](#) on page 254.

Below the snapshot view, a detailed information of each expression listed in the snapshot view is printed, as shown in [Figure 7-30](#) on page 352.

Figure 7-30 Detailed View of Evaluated Expressions



Identifiers assigned to min-term rows

SOP table

index	hit	<1>	<2>	
1.1.1	1	0	1	'&&' ←
1.1.2	0	1	0	
1.1.3	0	1	1	

index: 1.1 grade: 33% (1/3) line: 231 source:
 $(bist_cntrl_reg[0] == 1) \&& (bist_old_req == 0)$
 -----1-----> -----2----->

index: 1.2 grade: 50% (1/2) line: 231 source:
 $(bist_cntrl_reg[0] == 1)$
 -----3-----> -----4----->

index | hit | <3> | <4> |
 ----- | --- | --- | --- | '=='
 1.2.1 | 0 | lhs == rhs
 1.2.2 | 1 | lhs != rhs

index: 1.5 grade: 0% (0/2) line: 231 source:
 $(bist_old_req == 0)$
 -----5-----> -----6----->

index | hit | <5> | <6> |
 ----- | --- | --- | --- | '=='
 1.5.1 | 0 | lhs == rhs
 1.5.2 | 0 | lhs != rhs

This portion of the report prints the SOP table of each of the listed expression, in the snapshot view portion of the report. The `index` column in the SOP table prints the index as:

`<top-expr-id>.<sub-expr-id>.<min-term-row-id>`

where:

- `<top-expr-id>` is the identifier of the top-level expression.
- `<sub-expr-id>` is the identifier of the sub-level expression.
- `<min-term-row-id>` is the identifier of the each min-term row of the expression.

Example: \wedge and $\sim\wedge$ Operators (Exclusive OR and Equivalence, Greater than 4 terms)

Exclusive OR and bit-wise equivalence operations of more than four terms are scored using a parity tree. Instead of a truth table, the report contains a parity tree, in which, you typically find the following terms:

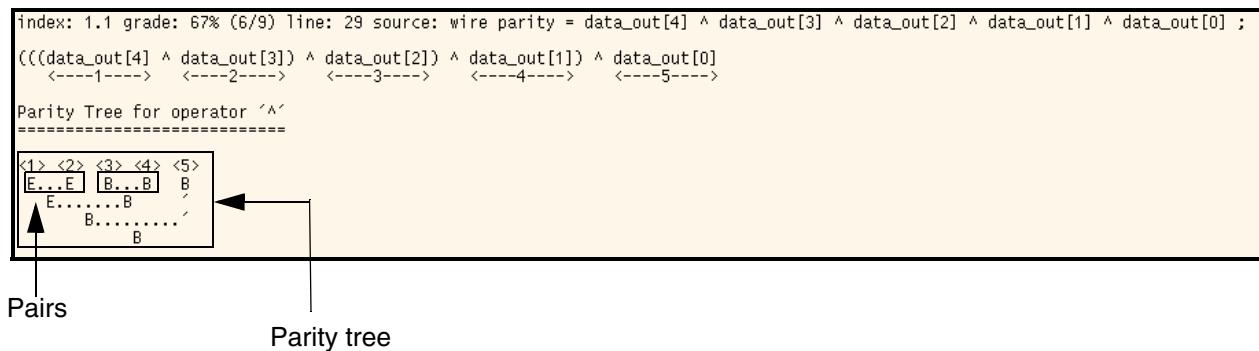
- E—Represents even bit
- O—Represents odd bit
- B—Represents both even and odd bits
- X—Represents no scoring of that bit
- I—Represents bits marked ignored

Scoring is 100% when each node of the tree has both (B). Any x or z input results in no scoring. The highest level in a parity tree represents each term in the expression individually. For example, five values at the highest level indicate five terms in the expression. A parity tree is formed by performing an exclusive OR between two expression terms (making a pair) at the highest level. The resultant output is the node at the next level. If the number of terms at the highest level is odd, the last term for which a pair could not be created is ignored for the time being and a pair is formed later when a similar situation is encountered at any of the lower levels. Consider the following expression.

```
wire parity = data_out[4] ^ data_out[3] ^ data_out[2] ^ data_out[1] ^ data_out[0];
```

[Figure 7-31 on page 353](#) shows parity tree created for this expression.

Figure 7-31 Parity Tree



Notice how the parity tree is created in this case.

- Highest level—E...E B...B B for each term in the expression.
- Second level—E..... B (This is the result after doing an exclusive OR between E...E and B...B at the highest level. The last B is ignored for the time being.)

- Third level— $B \dots \dots \dots'$ (This is the result after doing an exclusive OR between $E \dots \dots B$ at the second level.)
- Lowest level— B (This is the result after doing an exclusive OR between B from the third level and B from the highest level, which was ignored earlier.)

In the case of a parity tree, notice that the `index` column is not printed. As a result, the min-term row identifiers of the terms shown in a parity tree are not known. The min-term row identifiers are required at the time of excluding expressions from the parity tree. You can determine the min-term identifiers of a parity tree by appending

`<top-expr-id>.<sub-expr-id>` with `.<min-term-row-id>` for each term shown at all of the levels of a parity tree. Firstly, the min-term row identifiers are determined at the highest level of the parity tree. Subsequently, the min-term row identifiers are determined for the second level of the parity tree until the lowest level of the parity tree is reached. For example, in the above parity tree, `<top-expr-id>.<sub-expr-id>` is `1.1.1`. The min-term identifiers in this case are determined as:

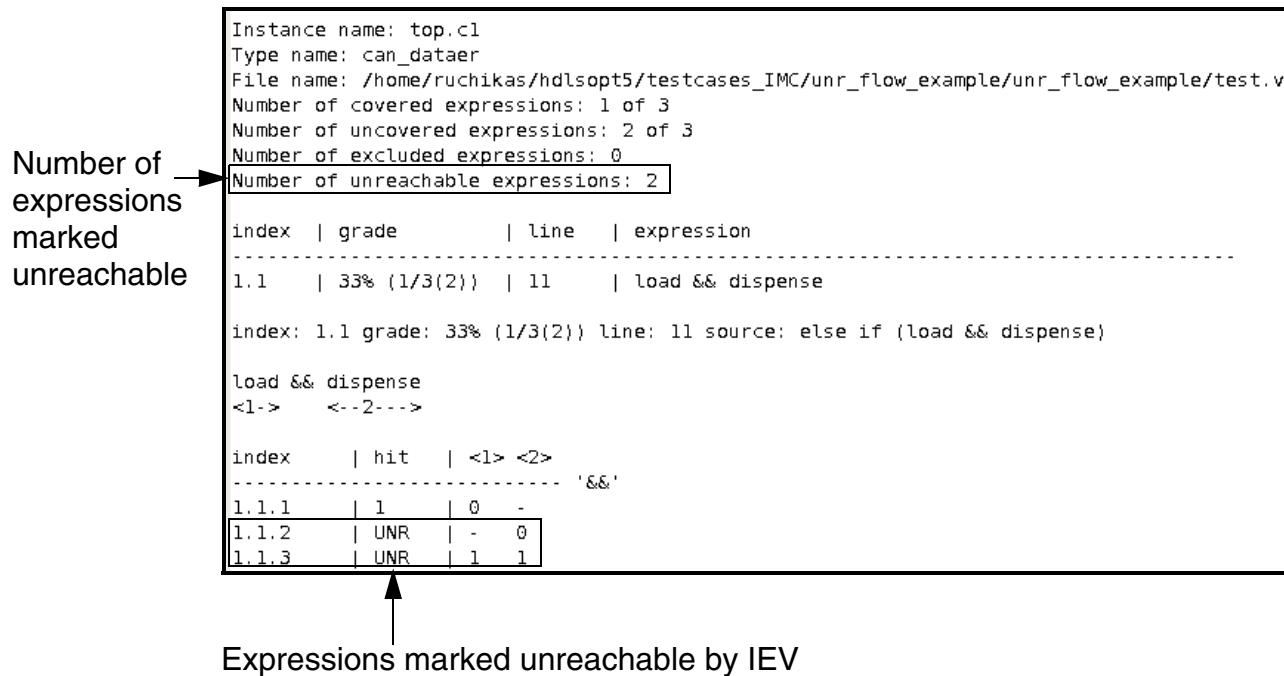
Level	Terms	Identified as
Highest level	$E \dots E B \dots B B$	<code>1.1.1, 1.1.2, 1.1.3, 1.1.4, and 1.1.5</code>
Second level	$E \dots \dots B$	<code>1.1.6 and 1.1.7</code>
Third level	B	<code>1.1.8</code>
Lowest level	B	<code>1.1.9</code>

For more information on interpreting results using optional control and vector scoring, see the Expression Coverage chapter of the *ICC User Guide*.

Example: Expressions Marked Unreachable by IEV

If an expression is marked unreachable by IEV, then the `Hit` column shows `UNR` for those expressions, as shown in [Figure 7-32](#) on page 355.

Figure 7-32 Instance-Based Detailed Report (with items marked UNR)



Number of expressions marked unreachable

Expressions marked unreachable by IEV

Instance name: top.cl Type name: can_dataer File name: /home/ruchikas/hdlsoft5/testcases_IMC/unr_flow_example/unr_flow_example/test.v Number of covered expressions: 1 of 3 Number of uncovered expressions: 2 of 3 Number of excluded expressions: 0 Number of unreachable expressions: 2			
index	grade	line	expression
1.1	33% (1/3(2))	11	load && dispense
index: 1.1 grade: 33% (1/3(2)) line: 11 source: else if (load && dispense)			
load && dispense	<1-> <-2--->		
index	hit	<1> <2>	
1.1.1	1	0	-
1.1.2	UNR	-	0
1.1.3	UNR	1	1

In the above report, the *Hit* column shows UNR for expressions 1.1.2 and 1.1.3 because they were determined as unreachable by IEV. In addition, the report header also shows the number of unreachable expressions.

Note: In the report, the grade is showing no decimal places because the value of `float_precision` configurable item is set to 0. By default, the value is set to 2 and you would see 2 decimal places in the attributes that show values as decimals. For more details, see [Setting Values for Configurable Items](#) on page 254.

Note: The UNR markers are also shown in the HTML-based report.

For details on the UNR flow and how items are marked UNR by IEV, see the *IEV User Guide*.

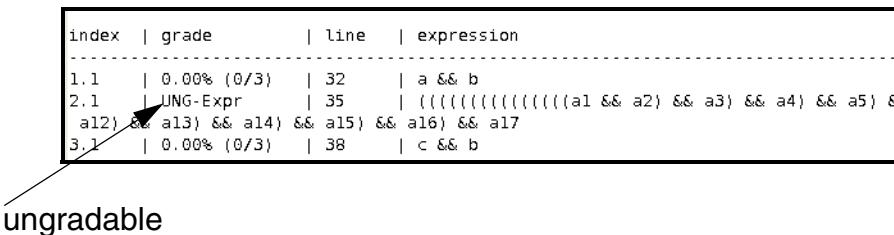
Example: Expressions Marked Ungradable

If an expression is marked ungradable at the time of simulation run, then the count columns shows UNG - Expr, as shown in [Figure 7-37](#) on page 360.

Figure 7-33 Instance-Based Detailed Report (with items marked UNG-Expr)

index	grade	line	expression
1.1	0.00% (0/3)	32	a && b
2.1	UNG-Expr	35	((((((((((a1 && a2) && a3) && a4) && a5) && a6) && a7) && a8) && a9) && a10) && a11) && a12) && a13) && a14) && a15) && a16) && a17
3.1	0.00% (0/3)	38	c && b

Items marked ungradable



In the above report, UNG - Expr is shown for the expressions determined as ungradable.

Note: The UNG - Expr markers are also shown in the HTML-based report.

Note: Merging between gradable and ungradable items is not allowed and appropriate warnings are reported in such cases.

For details on items marked ungradable, see the *ICC User Guide*.

Example: Toggle Coverage

To print a detail report displaying toggle coverage data for the instance ARB_INST, use:

```
report -detail -metrics toggle -both dtmf_recv_core.ARB_INST
```

Figure 7-34 on page 357 shows the output of the above command.

Figure 7-34 Toggle Coverage Detailed Report

Covered+Uncovered Toggle Detail Report, Instance Based			
=====			
Instance name: dtmf_recv_core.AR_B_INST			
Type name: arb			
File name: /home/ruchikas/cov_demo/dtmf/vlog_src/arb.v			
Number of covered signal bits: 12 of 15			
Number of uncovered signal bits: 3 of 15			
Number of signal bits partially toggled(rise): 0 of 15			
Number of signal bits partially toggled(fall): 0 of 15			
Hit(Full)	Hit(Rise)	Hit(Fall)	Signal

1	1	1	reset
1	1	1	clk
1	1	1	dma_breq
1	1	1	tdsp_breq
1	1	1	dma_grant
1	1	1	tdsp_grant
0	0	0	scan_in0
0	0	0	scan_en
0	0	0	scan_out0
1	1	1	next_state[2]
1	1	1	next_state[1]
1	1	1	next_state[0]
1	1	1	present_state[2]
1	1	1	present_state[1]
1	1	1	present_state[0]

Signal toggle information

In the above report,

- Covered signal bits indicates signals that have toggled through rise as well as fall transitions at least once.
- Uncovered signal bits indicates signals that have not toggled through rise as well as fall transitions even once.
- Signal bits partially toggled (rise) have toggled through rise transition only.
- Signal bits partially toggled (fall) have toggled through fall transition only.
- Hit(Full) is calculated as a minimum of Hit(Rise) and Hit(fall) counts.

Note: A partially toggled signal is considered as an uncovered signal by the merge precedence operation.

Example: Toggle Coverage (SystemVerilog Enumerated Toggles)

The toggle report also shows the SV enumerated toggles if it is enabled at the time of coverage data collection. By default, SV enumerated toggles are not collected. The collection of enumerated toggles is enabled using CCF command `set_toggle_scoring -sv_enum`. For more details on this command, see the *ICC User Guide*.

Figure 7-35 on page 358 shows a toggle coverage report with SV enumerated toggles.

Figure 7-35 Toggle Coverage Detailed Report

The figure displays a detailed toggle coverage report. At the top is a header table with various metrics. Below it is a main table for classic toggles, and at the bottom is a separate table for enum toggles. Arrows point from labels to each table.

Header Table (Enum toggle information):

```

Instance name: top
Type name: top
File name: /vobs/mac_test/test/mac_batch_test/test_designs/
Number of covered toggles: 9 of 17
Number of uncovered toggles: 8 of 17
Number of excluded toggles: 0
Number of unreachable toggles: 0
Number of covered signal bits: 7 of 13
Number of uncovered signal bits: 6 of 13
Number of excluded signal bits: 0
Number of unreachable signal bits: 0
Number of signal bits partially toggled(rise): 1 of 13
Number of signal bits partially toggled(fall): 0 of 13
Number of covered enum toggles: 2 of 4
Number of uncovered enum toggles: 2 of 4
Number of excluded enum toggles: 0
Number of unreachable enum toggles: 0

```

Main Table for classic toggles:

Hit(Full)	Hit(Rise)	Hit(Fall)	Signal
1	1	1	a[3]
1	1	1	a[2]
0	0	0	a[1]
0	1	0	a[0]
1	1	1	d[3]
1	1	1	d[2]
1	1	1	d[1]
1	1	1	d[0]
1	1	1	clk
0	0	0	my_bit[1]
0	0	0	my_bit[0]
0	0	0	my_logic[1]
0	0	0	my_logic[0]

Table for enum toggle:

Hit	Value	Enum
0	green	color
1	red	
1	yellow	
0	blue	

In the above report,

- A separate table is created for enum toggles.
- The total count of toggles includes both enum as well as classic toggles. In this case, it is 17 because 13 are classic toggles and 4 are enum toggles.
- The report header shows information of enum toggles as:
 - Number of covered enum toggles: This indicates the number of hit enum values out of total number of enum values.
 - Number of uncovered enum toggles: This indicates the number of uncovered enum values out of total number of enum values.

- ❑ Number of excluded enum toggles: This indicates the number of excluded enum values.
- ❑ Number of unreachable enum toggles: This indicates the number of unreachable enum values.

Note: The enum toggles are also shown in the HTML-based report. In addition, the merge operation and the rank operation also support enumerated toggles.

Example: Signals Marked Unreachable by IEV

If a signal is marked unreachable by IEV, then the *Count* column shows UNR, as shown in [Figure 7-36 on page 359](#).

Figure 7-36 Instance-Based Detailed Report (with items marked UNR)

Hit(Full)	Hit(Rise)	Hit(Fall)	Signal
1	1	1	clk
0	0	1	reset
UNR	UNR	UNR	load
UNR	UNR	UNR	dispense
0	0	0	data[1]
0	0	0	data[0]
UNR	UNR	0	left[1]
UNR	UNR	0	left[0]

In the above report,

- The report header shows the number of unreachable toggles.
- The Hit(Rise) and Hit(fall) columns show UNR if the item is determined as unreachable by IEV.
- Hit(Full) column shows UNR if at least one or both (rise and fall) transitions are marked unreachable by IEV.

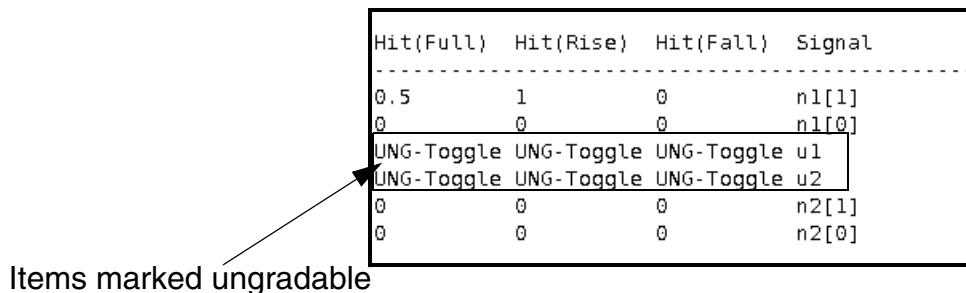
Note: The UNR markers are also shown in the HTML-based report.

For details on the UNR flow and how items are marked UNR by IEV, see the *IEV User Guide*.

Example: Signals Marked Ungradable

If a toggle variable is marked ungradable at the time of simulation run, then the count columns shows UNG, as shown in [Figure 7-37](#) on page 360.

Figure 7-37 Instance-Based Detailed Report (with items marked UNG)



Hit(Full)	Hit(Rise)	Hit(Fall)	Signal
0.5	1	0	n1[1]
0	0	0	n1[0]
UNG-Toggle	UNG-Toggle	UNG-Toggle	u1
UNG-Toggle	UNG-Toggle	UNG-Toggle	u2
0	0	0	n2[1]
0	0	0	n2[0]

In the above report, Hit(Full), Hit(Rise), and Hit(fall) columns show UNG – Toggle for the items determined as ungradable.

Note: The UNG – Toggle markers are also shown in the HTML-based report.

Note: Merging between gradable and ungradable items is not allowed and appropriate warnings are reported in such cases.

For details on items marked ungradable, see the *ICC User Guide*.

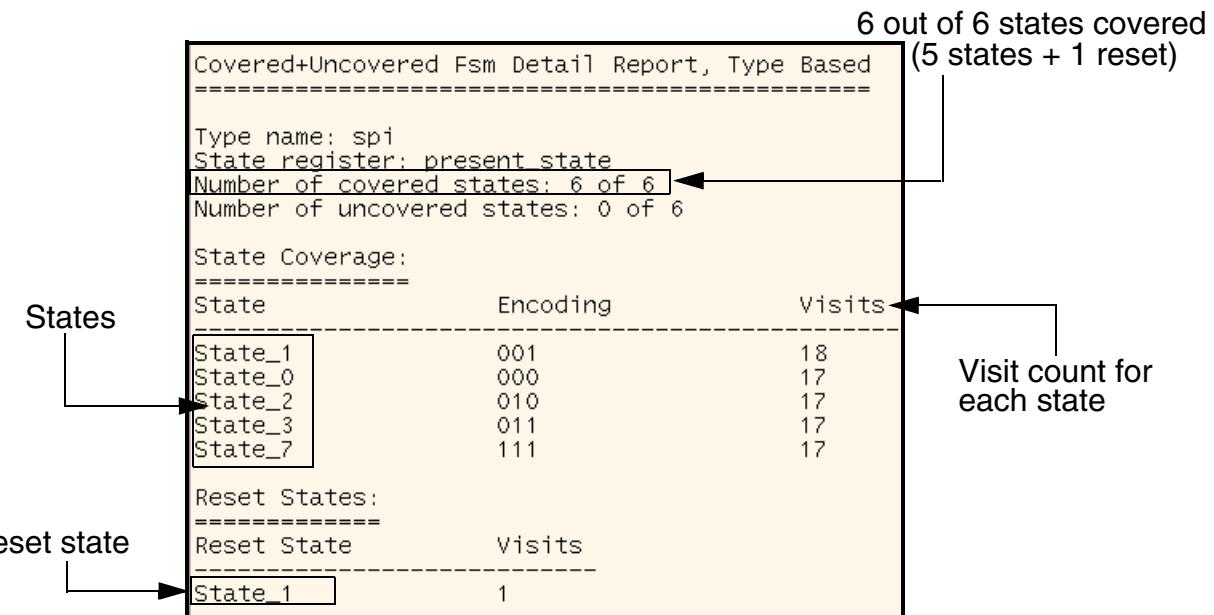
Example: FSM State Coverage

To report detailed state coverage data for the module spi, as shown below, use:

```
report -detail -metrics state -both -type spi
```

[Figure 7-38](#) on page 361 shows the output of the above command.

Figure 7-38 FSM State Coverage Detailed Report



By default, reset states are not considered in calculating the number of states. To consider reset states in coverage numbers, use the `set_fsm_reset_scoring` command in the CCF and regenerate coverage data. The above report is generated with `set_fsm_reset_scoring` enabled in the CCF.

In the absence of this command, the number of states would be 5 instead of 6, and number of visits for state `State_1` would be 19.

Note: To exclude a reset state or transition, `_RST` should be appended to the state name. The report above shows reset state as `State_1`. In case you want to exclude this state, then the state name must be specified as `State_1_RST`, as shown below:

```
exclude -type spi -state present_state.State_1_RST
```

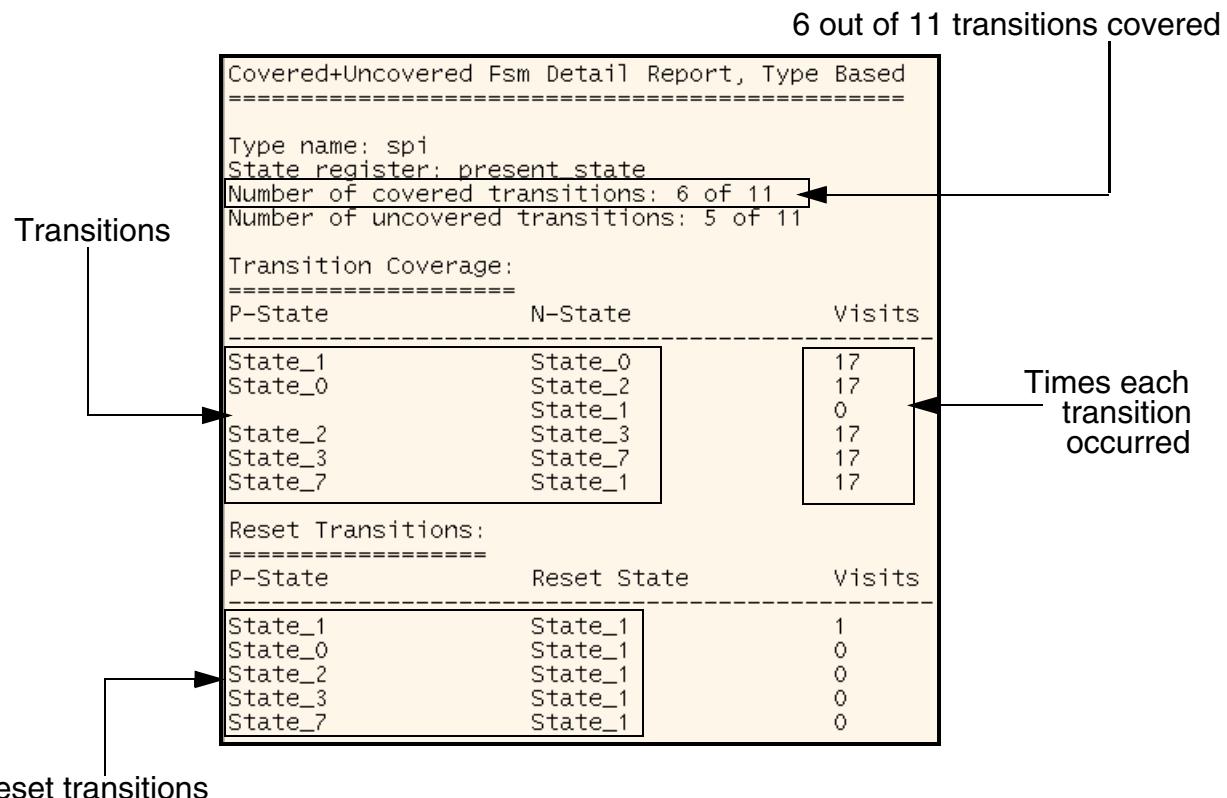
Example: FSM Transition Coverage

To report detailed transition coverage data for the module `spi`, use:

```
report -detail -metrics transition -both -type spi
```

[Figure 7-39](#) on page 362 shows the output of the above command.

Figure 7-39 FSM Transition Coverage Detailed Report



By default, reset transitions are not considered in calculating the number of transitions. To consider reset transitions in coverage numbers, use the `set_fsm_reset_scoring` command in the CCF and regenerate coverage data. The above report is generated with `set_fsm_reset_scoring` enabled in the CCF.

In the absence of this command, total number of transitions would be 6 instead of 11 (excluding 5 reset transitions).

Note: When excluding a reset state or transition, `_RST` should be appended to the state name. The report above shows reset state as `State_1`. In case you want to exclude this transition, then the state name must be specified as `State_1_RST`, as shown below:

```
exclude -type spi -transition present_state.State_0.State_1_RST
```

Note: By default, ICC does not score FSM hold transitions (`State_0 -> State_0`). To enable scoring of FSM hold transitions, use the `set_fsm_scoring -hold_transition` in the CCF, and regenerate coverage data.

Example: FSM Arc Coverage

To report arcs, first enable arc scoring for selected FSMs, using the `set_fsm_arc_scoring` command in the CCF and regenerate coverage data. After enabling arc scoring, each state transition with all possible input conditions under which the transition takes place is reported.

To report detailed arc coverage data for the module `spi`, use:

```
report -detail -metrics arc -both -type spi
```

[Figure 7-40 on page 363](#) shows the output of the above command.

Figure 7-40 FSM Arc Coverage Detailed Report

Covered+Uncovered Fsm Detail Report, Type Based

Type name: `spi`
 State register: `present_state`
 Number of covered arcs: 6 of 13
 Number of uncovered arcs: 7 of 13

Arc Coverage:

Inputs = (`spi_fs`, `spi_clk`, `not_full`, `full`)

P-State	N-State	Inputs	Visits
<code>State_1</code>	<code>State_0</code>	10--	17
<code>State_0</code>	<code>State_2</code>	10--	17
	<code>State_1</code>	-1--	0
		0---	0
<code>State_2</code>	<code>State_3</code>	-1--	17
		--0-	0
<code>State_3</code>	<code>State_7</code>	-0-1	17
<code>State_7</code>	<code>State_1</code>	----	17

Reset Transitions:

P-State	Reset State	Visits
<code>State_1</code>	<code>State_1</code>	1
<code>State_0</code>	<code>State_1</code>	0
<code>State_2</code>	<code>State_1</code>	0
<code>State_3</code>	<code>State_1</code>	0
<code>State_7</code>	<code>State_1</code>	0

Arcs for each transition

Evaluated covered because one of the arcs is covered

The Input column lists all possible arcs for each transition. In the above report, transition `State_2` -> `State_3` happens under two input conditions, `-1--` and `--0-`, respectively.

A transition is evaluated covered if any of the arcs for that transition are covered. A transition is evaluated uncovered if all of the arcs for that transition are uncovered. In the above report,

transition State_0 → State_1 is considered uncovered because both of the arcs for this transition are uncovered.

Note: The number of visits for a state may vary when arc coverage is enabled versus when arc coverage is disabled. If any of the inputs is x, it will match the corresponding column in the SOP row only if it is -. It will not match 1/0. If the input values do not match any SOP row because of any input being x, ICC will not increment state and transition counts for that transition. If arc scoring is disabled, ICC does not match any SOP rows and counts the transition.

Examples: Assertion Coverage

To generate a detailed report for assertion coverage items for instance DMA_INST in module dtmf_recv_core, use:

```
report -detail -metrics assertion -both dtmf_recv_core.DMA_INST
```

Figure 7-41 on page 364 shows the output of the above command.

Figure 7-41 Assertion Coverage Detailed Report

3 out of 7 assertions covered

Finished	Failed	Assertion	Line	Source Code
0	17	dma_req_when_data_ready	165	// ps1 assert dma_req_when_data_ready;
17	0	dma_wait_for_grant	172	// ps1 assert dma_wait_for_grant;
0	17	dma_read_spi_pulse	187	// ps1 assert dma_read_spi_pulse;
0	17	dma_address_strobe_pulse	190	// ps1 assert dma_address_strobe_pulse;
17	0	dma_transfer_reset_delay	199	// ps1 assert dma_transfer_reset_delay;
0	0	dma_bufselect1	205	// ps1 assert dma_bufselect1;
4	0	dma_bufselect0	208	// ps1 assert dma_bufselect0;

Covered+Uncovered Assertion Detail Report, Instance Based

=====
Instance name: dtmf_recv_core.DMA_INST
Type name: dma
File name: /home/ruchikas/cov_demo_dtmf/vlog_src/dma.v
Number of covered assertions: 3 of 7
Number of uncovered assertions: 4 of 7

Assertion successfully finished 4 times

To generate a detailed report for assertion coverage items (along with the status of each assertion) for instance ARB_INST in module dtmf_recv_core, use:

```
report -detail -metrics assertion -all dtmf_recv_core.ARST -assertionStatus
```

Figure 7-42 on page 365 shows the output of the above command.

Figure 7-42 Assertion Coverage Detailed Report (with Status Information)

Assertions status					
Instance name:	Type name:	File name:	Number of covered assertions:	Number of uncovered assertions:	Number of excluded assertions:
dtmf_recv_core.AR_B_INST	arb	/home/ruchikas/cov_demo_dtmf/vlog_src/arb.v	8 of 11	3 of 11	0
Finished	Failed	Status	Assertion	Line	Source Code
-----	-----	-----	-----	-----	-----
0	0	Other	__cover9_arb_dmar_dropped	201	// psl cover arb_dmar_dropped;
0	0	Other	__cover8_arb_tdspr_dropped	198	// psl cover arb_tdspr_dropped;
2	0	Passed	__cover7_arb_tworeqs	195	// psl cover arb_tworeqs;
117	0	Passed	__cover6_arb_dmar	193	// psl cover arb_dmar;
265	0	Passed	__cover5_arb_tdspr	191	// psl cover arb_tdspr;
52864	16	Failed	__sugar_assert_arb_never_grant_wo_req_4	185	// psl assert arb_never_grant_wo_req(tdsp_breq.tdsp_grant);
53028	0	Passed	__sugar_assert_arb_never_grant_wo_req_3	184	// psl assert arb_never_grant_wo_req(dma_breq.dma_grant);
262	0	Passed	__sugar_assert_arb_hold_grant_2	167	// psl assert arb_hold_grant(tdsp_breq.tdsp_grant);
68	0	Passed	__sugar_assert_arb_hold_grant_1	166	// psl assert arb_hold_grant(dma_breq.dma_grant);
53150	0	Passed	arb_never_two	159	// psl assert arb_never_two;
0	0	Other	arb_tdspr_priority	175	// psl assert arb_tdspr_priority;

In the above report, an additional column **Status** is added and the possible values in the **Status** column can be:

- **Passed** if the assertion finished without failure at least once, and never failed.
- **Failed** if the assertion finished with failure at least once.
- **Other** if the assertion never finished.

Consider another example, where **-abvrecordvacuous** switch is used at the time of simulation run. In such cases, you can enable reporting of Vacuous, Attempt, and Disabled counters in addition to the default Finished and Failed counters.

For example, to generate a detailed assertion coverage report and display the Vacuous, Attempt, and Disabled counters, use:

```
report -detail -metrics assertion -all -assertionStatus -allAssertionCounters
```

[Figure 7-43](#) on page 366 shows the output of the above command.

Figure 7-43 Assertion Coverage Detailed Report

Additional counters (due to use of `-allAssertionCounters` option)

Finished	Failed	Vacuous	Attempt	Disabled	Status	Assertion	Line	Source Code
2	0	1	4	1	Passed	L14	143	
0	0	0	1	1	Disabled	L3	140	
0	0	0	0	0	Not Attempted	L19	137	
3	1	0	4	1	Failed	L12	136	
0	1	0	1	1	Failed	L10	135	
2	7	0	9	0	Failed	L2	134	
0	0	0	9	0	Attempted	L18	131	
0	9	0	9	0	Failed	L17	130	
5	0	1	9	0	Passed	L16	129	
1	1	1	4	1	Failed	L15	128	
0	2	1	4	1	Failed	L13	127	
1	0	0	1	1	Passed	L11	126	
0	0	1	1	1	Vacuous	L9	125	
0	4	1	9	0	Failed	L8	124	
9	0	0	9	0	Passed	L7	123	
0	0	4	9	0	Vacuous	L4	122	
3	2	1	9	0	Failed	L1	121	

Status field also shows more detailed status
(Vacuous, Attempted, Not Attempted, Disabled)

In the above report, following items are shown (due to the use of `-allAssertionCounters` option):

- Additional columns Vacuous, Attempt, and Disabled are reported to show their respective counters.
- The *Status* column (in addition to Passed and Failed values) shows following items:
 - Vacuous if the assertion was vacuous pass at least once but never finished.
 - Disabled if the assertion transitioned to the disabled state at least once but never finished.
 - Attempted if the assertion was attempted at least once but never finished.
 - Not Attempted if the assertion was not attempted all.

Consider another example, to generate a detailed assertion coverage report to display formal properties. For this, use:

```
report -detail -metrics assertion -all -aspect formal -assertionStatus
```

[Figure 7-43 on page 366](#) shows the output of the above command.

Figure 7-44 Assertion Coverage Detailed Report

Formal properties

Formal	Finished	Formal Failed	Formal Status	Assertion	Line	Source Code
1	0	0	Proved	ASSERT_PASS_TRACE_PASS	39	
UNR	UNR	0	Other	ASSERT_PASS_TRACE_FAIL	40	
1	0	0	Failed	ASSERT_FAIL_TRACE_FAIL	41	
1	0	0	Failed	ASSERT_FAIL_TRACE_PASS	42	
1	0	0	Other	COVER_PASS	43	
UNR	UNR	0	Other	COVER_FAIL	44	

Formal attributes reported

In the above report, formal attributes (*Formal Finished*, *Formal Failed*, and *Formal Status*) are shown. This is because, `-aspect formal` is specified on the command line.

Example: Covergroup Coverage

To display a detail report for covergroup coverage for type `tdsp_core`, use:

```
report -detail -metrics covergroup -both -type tdsp_core
```

[Figure 7-45](#) on page 368 shows the output of the above command.

Figure 7-45 Covergroup Coverage Detailed Report

Covergroup name	Coverpoint name	Average and Covered grade
<pre>Type name: tdsp_core File name: /home/puchikas/cov_demo_dtmf/vlog_src/tdsp_core.v Number of covered cover bins: 10 of 56 Number of uncovered cover bins: 46 of 56</pre>		
	Name	Average, Covered Grade
	cg	49.71%, 18.00% (10/56)
	-A	100.00% (2/2)
	-scalar_min	100.00% (358/3)
	-scalar_max	100.00% (36/3)
	-A2	28.00% (4/14)
	-vector_low[18]	100.00% (42/40)
	-vector_low[19]	100.00% (45/40)
	-vector_low[20]	100.00% (42/40)
	-vector_med[20]	100.00% (42/40)
	-vector_med[21]	0.00% (36/40)
	-vector_med[22]	0.00% (36/40)
	-vector_med[23]	0.00% (15/40)
	-vector_med[24]	0.00% (0/40)
	-vector_med[25]	0.00% (6/40)
	-vector_med[26]	0.00% (0/40)
	-vector_med[27]	0.00% (0/40)
	-vector_med[28]	0.00% (0/40)
	-vector_med[29]	0.00% (0/40)
	-vector_med[30]	0.00% (0/40)
	-B	20.00% (2/10)
	-auto[0:429496728]	100.00% (17314/1)
	-auto[429496729:858993457]	100.00% (3/1)
	-auto[858993458:1288490186]	0.00% (0/1)
	-auto[1288490187:1717986915]	0.00% (0/1)
	-auto[1717986916:2147483644]	0.00% (0/1)
	-auto[2147483645:2576980373]	0.00% (0/1)
	-auto[2576980374:3006477102]	0.00% (0/1)
	-auto[3006477103:3435973831]	0.00% (0/1)
	-auto[3435973832:3865470560]	0.00% (0/1)
	-auto[3865470561:4294967295]	0.00% (0/1)
	-B2	100.00% (2/2)
	-opbmin	100.00% (565/1)
	-opbmax	100.00% (9/1)
	- AXA2	0.00% (0/28)
<p>Bins</p> <p>Cross between coverpoints A and A2</p>		
		Line Source Code
		187 covergroup cg @{negedge clk};
		191 A: coverpoint TDSP_CORE_GLUE_INST.opb;
		196 bins scalar_min = {0};
		197 bins scalar_max = {10};
		201 A2: coverpoint TDSP_CORE_GLUE_INST.opb;
		206 bins vector_low[] = {[18:20]};
		206 bins vector_low[] = {[18:20]};
		206 bins vector_low[] = {[18:20]};
		207 bins vector_med[] = {[20:30]};
		207 bins vector_med[] = {[20:30]};
		207 bins vector_med[] = {[20:30]};
		207 bins vector_med[] = {[20:30]};
		207 bins vector_med[] = {[20:30]};
		207 bins vector_med[] = {[20:30]};
		207 bins vector_med[] = {[20:30]};
		207 bins vector_med[] = {[20:30]};
		207 bins vector_med[] = {[20:30]};
		207 bins vector_med[] = {[20:30]};
		207 bins vector_med[] = {[20:30]};
		207 bins vector_med[] = {[20:30]};
		207 bins vector_med[] = {[20:30]};
		212 B: coverpoint TDSP_CORE_GLUE_INST.opb;
		216 B2: coverpoint TDSP_CORE_GLUE_INST.opb;
		220 bins opbmin = {0};
		221 bins opbmax = {60};
		211 AXA2: cross A, A2;

In the above report:

- *Name* column displays the names of covergroups, coverpoints, crosses, and bins.
- *Average.Covered Grade* column shows the average as well as covered grade for each coverage item in the design. For details on grade calculation, see [Coverage Grading in IMC](#).

Note: In a detailed report, a default bin, if covered (with hit count > 1) is reported. Displaying default bins helps users to keep track of erroneous values or any uncaptured values by any other bin. The covered default bins though reported are ignored while calculating the total number of bins. The following items are not reported in a detailed report:

- Uncovered default bins

- Uncovered bins if the total number of bins for a coverpoint reaches the limit of 0xfffffff with a particular bin. For example,

```
A: coverpoint a{  
    bins b1[] = {5,8};  
    bins b2[] = {$:4};  
    bins b3[] = {7,6};  
    bins b4[] = {9,11};  
}
```

For the above code, if the total number of bins reaches the limit of 0xfffffff with bin b2, then uncovered bins for b2 and bins defined after b2 (b3 and b4) will not be reported.

In the above report, each bin within a coverpoint or cross is printed separately. You can disable printing of bin information using the following command:

```
report -detail -metrics covergroup -both -type tdsp_core -kind abstract
```

[Figure 7-46 on page 369](#) shows the output of the above command.

Figure 7-46 Covergroup Coverage Detailed Report (without bin information)

Covered+Uncovered CoverGroup Detail Report, Type Based				
=====				
Type name: tdsp_core				
File name: /home/ruchikas/cov_demo_dtmf/vlog_src/tdsp_core.v				
Number of covered cover bins: 10 of 56				
Number of uncovered cover bins: 46 of 56				
Name	Average, Covered Grade	Line	Source	Code
cg	49.71%, 18.00% (10/56)	187	covergroup cg @{negedge clk};	
- A	100.00% (2/2)	191	A: coverpoint TDSP_CORE_GLUE_INST.opa	
- A2	28.00% (4/14)	201	A2: coverpoint TDSP_CORE_GLUE_INST.opa	
- B	20.00% (2/10)	212	B: coverpoint TDSP_CORE_GLUE_INST.opb	
- B2	100.00% (2/2)	216	B2: coverpoint TDSP_CORE_GLUE_INST.opb	
-- AXA2	0.00% (0/28)	211	AXA2: cross A, A2;	

Bin information not printed in the detailed report

Important

The detailed report for block, expression, and functional coverage displays the line description information from the source file used to generate coverage data. Line description information is not displayed if the location or content of the source file changes. If the location changes, use the `sourcemap` command to map paths so that IMC can locate the source files. If the content changes, a warning is reported. For details on the `sourcemap` command, see [Mapping Source Path on page 247](#).

Note: The overflow count of a coverage item is 2147483647. At times, when thousands of coverage databases are merged, the score of a coverage item might exceed the overflow

count. In such cases, the score of that coverage item is shown as the overflow count, which is 2147483647.

Example: Covergroup Coverage (Excluded Items)

To display a covergroup detail report with excluded items, use:

```
report -detail -metrics covergroup -excludes
```

Figure 7-47 on page 370 shows the output of the above command.

Note: The report with excluded item also shows ungradable items, if any.

Figure 7-47 Covergroup Coverage Detailed Report

```
Excluded CoverGroup Detail Report, Instance Based
=====
Instance name: test
Type name: test
File name: /vobs/mac_test/test/mac_batch_test/test_designs/sanity/html_reporting/compact/ungradeable/test.sv
Number of excluded cover bins: 500625

Name          Average, Covered Grade    Line  Source Code
-----
c1           8%, 7% (43/570/500625)    18
| --A          UNG                     21
| | --b1[1]    EXCL                   22
```

In the above report, both excluded as well as ungradable items are reported.

7.2.1.4 HTML Reports

Coverage data can also be analyzed in a web browser. To generate reports in an HTML format use the `-html` option of `report` command. The reports generated in an HTML format includes hyperlinks to navigate the design hierarchy and covergroup items, view cumulative and self coverage numbers, view detailed reports for specific coverage types, and view relevant source code.

HTML reports can be viewed in any standard web browser, such as Internet Explorer, Firefox4 and above and so on.

Note: Reports in HTML format can also be generated using the `report_metrics` command. The look and feel of the HTML report generated with the `report -html` command and the `report_metrics` command is different. The report generated with `report_metrics` command is aligned with the reports generated from IMC GUI.

Generating Reports in an HTML Format Using report -html Command

To generate reports in an HTML format, use:

```
report [-summary | -detail]
[-type | -inst]
[-metrics <metrics_type>]
-html
[-out <name>]
[-overwrite]
[-grading <average> | <covered> | <both>]
[-cross <expand> | <aggregate>]
[-kind <expand> |
  <compact> [-cubeExpand on|off] |
  <aggregate> [[-atleast <num> | -upto <num>] [-cubesize <min>]
  [-cubestars <min>] [-excludeNA] [-illegalIgnore]]
[-showempty on|off]
[-covered | -uncovered | -excludes | -all | -both]
[-source on|off]
[-exclComments]
[-aspect sim | formal | both]
[-assertionStatus]
[-allAssertionCounters]
<list>
```

where

```
metrics_type ::= [overall] [all] [code] [fsm] [functional] [block] [expression]
[toggle] [state] [transition] [arc] [covergroup] [assertion]
```

Note: The `-kind` and `-cross` options are valid only with the `-detail` option.

In the above BNF

- `-summary | -detail` specifies the type of report that must be generated in an HTML format. If not specified, `-detail` is assumed.
- `-html` generates summary or detail report in an HTML format.
- `-out <name>` redirects the HTML report files to an alternate location instead of the default `html_<timestamp>` directory.

If the `-out` option is not specified, by default, a directory named `html_<timestamp>` is created in the current working directory, and coverage HTML reports along with the top-level summary page (`index.html`) are stored in the `html_<timestamp>` directory.

Note: A top-level summary page `index.html` is used to navigate through the HTML reports available in the `html_<timestamp>` directory.

With the `-out` option, the command redirects the HTML output to `<name>`. If `<name>` includes just the name of the output directory, then the command creates the output directory named `<name>` in the current working directory, and stores the HTML report

files along with the top-level summary page in <name>. For example, if <name> is specified as:

```
report -detail -html -all -type * -out day1
```

then directory named day1 is created in the current working directory to store HTML report files.

Note: If <name> already exists, the output directory is not overwritten. You must use the -overwrite option to overwrite an existing output directory.

If <name> includes the complete path, then the command creates a directory by the name mentioned at the lowest level in the path, and stores the HTML report files at that location. For example, if <name> is specified as:

```
report -detail -html -all -type * -out data/myreports/day1
```

then directory named day1 is created under data/myreports to store HTML report files. In this case, the path (data/myreports) must exist. The directory day1 must not exist, as it will be created and not overwritten unless the -overwrite option is used.

- -overwrite enables overwriting of the existing HTML output directory. By default, HTML output directory is not overwritten and an error is reported.
- -grading <average> | <covered> | <both> enables printing of average grade, covered grade, or both (average as well as covered) grades in the report. By default, both covered as well as average grades are printed in the HTML report.

To print only the average grade in the HTML report, use:

```
report -detail -html -all -out detail_rep -inst *... -grading average
```

To print only the covered grade in the HTML report, use:

```
report -detail -html -all -out detail_rep -inst *... -grading covered
```

Note: To print both (covered as well as average) grades, either use -grading both or do not use the -grading option. In the absence of the -grading option, both is assumed.

- -cross <expand> | <aggregate> specifies the format in which the automatically generated cross bins must be printed in the HTML report. The format can be specified as any of the following:
 - expand—to print the cross tuple information in a list format. This is the default behavior.
 - aggregate—to print the cross tuple information in a tabular format after grouping similar bins.

For more details, see [Example: Cross Information in HTML Reports](#) on page 384.

- `-kind` specifies the format in which the covergroup bin information must be printed in the HTML report. The format can be specified as any of the following:
 - `expand`—to print the bin information as a list without grouping similar bins. This is the default behavior.
 - `compact`—to print the bin information after grouping similar bins (cubes). For each cube, a separate detailed page is created which includes information about the bins included in that cube. The compact option automatically turns OFF in some cases. For more details, see [Important points related to -kind compact option](#) on page 391.
 - `-cubeExpand on|off` enables or disables generation of separate detailed page for each cube. By default, the value of this option is set to `on`, which enables generation of detailed information page for each cube. To disable generation of detailed information page, set the `-cubeExpand` option to `off`. Setting this option as `off` is useful when the information on the detailed page is not of much relevance and also in situations where you want to save the disk space.
 - `aggregate`—to print the cross tuple information in a tabular format after grouping similar bins. While generating aggregated results, you can specify following options:
 - `-atleast <num> | -upto <num>` — to set the minimum and maximum number of samples required for a bin to be considered covered.
 - `-cubesize <min>` — to set the minimum size of reported cubes, (measured in % of physical space size).
 - `-cubestars <min>` — to set the minimum number of stars (*) in the reported cubes. Stars are used in aggregation results automatically when it makes the results more readable.
 - `-excludeNA` — If specified, NA bins are not included in reported holes/covered cubes.
 - `-illegalIgnore` — to include aggregation also for illegal and ignore space.

Note: The `-kind` option is supported only with HTML reports.

For more details, see [Example: Bin Information in HTML Reports](#) on page 387.

See [Summary Report](#) and [Detailed Report](#) for details on syntax description of other options.

For example, to generate HTML reports of all metrics types for all of the instances in the design, use:

```
report -detail -html -all -out detail_rep -overwrite -inst *...
```

Incisive Metrics Center User Guide

The above command creates a directory named `detail_rep` in the current working directory (if it does not exist, else it is overwritten) and creates following files and folders in it.

File/Directory Name	Description
<code>index.html</code>	It is the top-level summary page using which you can navigate through the HTML reports. You must open this page in the standard web browser to navigate through the HTML reports.
<code>global_cg_summ.html</code>	This file displays the global covergroup summary using which you can navigate through the covergroup items scored in the design. This file is generated only if a detailed HTML report is generated.
<code>legend.html</code>	This is the help page of HTML reports. It shows the detailed description of various columns and items shown in the HTML report.
<code>report_sub_dir</code>	This directory has various subdirectories, such as <code>dir_1</code> , <code>dir_2</code> , <code>dir_3</code> and so on to store individual reports for different instances or types specified in the <code>report</code> command.

Incisive Metrics Center User Guide

File/Directory Name	Description
list_refinement	<p>This file includes the details of the refinements applied before generating the HTML report.</p> <ul style="list-style-type: none">■ If refinements are applied using the <code>exclude/unexclude</code> commands before generating the HTML report, the <code>list_refinement</code> file includes the following note: Note: A few <code>exclude/unexclude</code> commands were executed before generating the HTML report. Those commands, though considered while generating the report, are not listed here.■ If refinements are applied by loading the refinement files, then the <code>list_refinement</code> file includes the list of those refinement files.■ If refinements are applied using both <code>exclude/unexclude</code> commands and by loading the refinement files, then along with the list of refinement files the following note is printed in the <code>list_refinement</code> file: Note: In addition to the refinements mentioned in the below files, a few <code>exclude/unexclude</code> commands were also executed before generating the HTML report. Those commands, though considered while generating the report, are not listed here.■ If no refinements are applied before generating the HTML report, the <code>list_refinement</code> file includes the following text: There were no refinement files applied
list_ccf	This file includes the details of the coverage configuration files (CCF) used at the time of coverage data generation. The file includes the list of CCF files as well as the contents of the CCF files.

Note: The `report -html` command generates the above files and folders. However, at times, you might notice a few additional files/directories in the HTML output directory. These additional files/directories are found in the HTML output directory if after the `report -html` command, `rank` command is used to generate ranking results in the same output directory.

The `-attach on` option of the `rank` command links the ranking output with the HTML report and therefore, additional files/directories are created in the HTML output directory. For more details, see [Example: Use of -attach option](#) on page 313 of [Chapter 6, “Ranking Runs”](#).

Viewing HTML Reports

Consider an example of an instance-based detailed HTML report generated using:

```
report -detail -html -out detail_rep -overwrite -metrics
code:block:expression:toggle:fsm:functional -inst *...
```

Note: The above command generates a report that includes metric types code, block, expression, toggle, FSM, and functional. You can include more metric types by separating them with a colon.

To view HTML reports, open `detail_rep/index.html` in a standard web browser.

[Figure 7-48](#) on page 376 shows the top-level summary page `index.html` in the browser.

Figure 7-48 Top-Level Summary Page

Top-level summary report

Top-level summary page

Click to view report legend and help

Click to view covergroup summary

Table color legend

Click to view the CCF files used at the time of coverage data generation

Click to view the list of refinement files applied

The top-level summary page:

- Displays the [Coverage Top-Level Summary Report](#) and

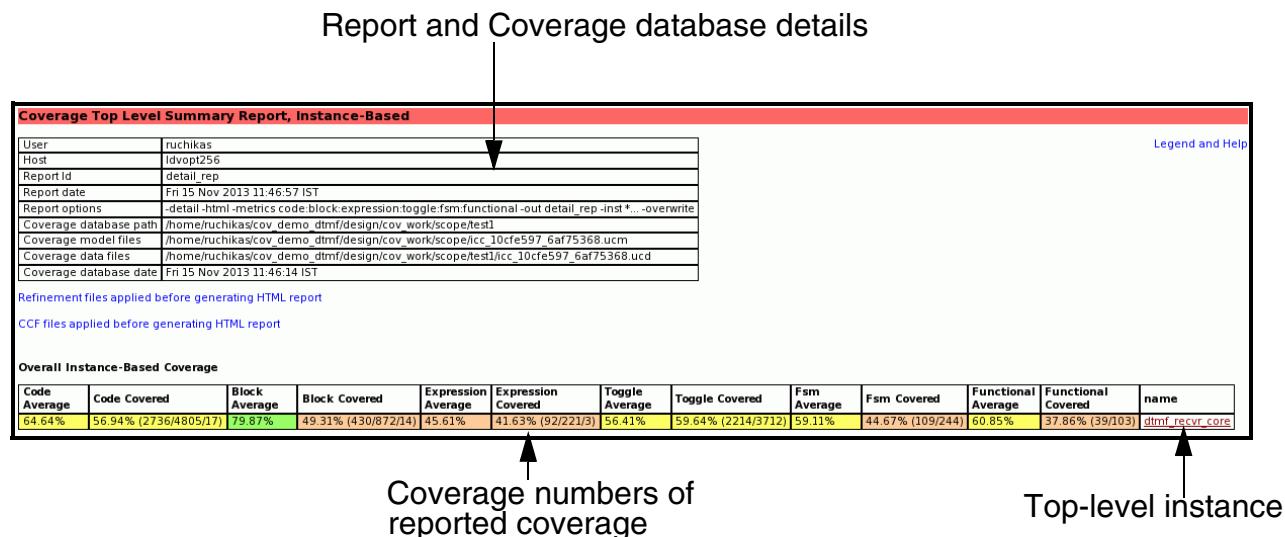
- Provides a link to access the [Global Covergroup Summary](#)

Coverage Top-Level Summary Report

The *Coverage Top-level Summary Report* section displays information, such as who generated the report, when the report was generated, the options used to generate the report, coverage database location, model file (.ucm) and data file (.ucd) used to generate the report, and time when the coverage database was generated. The report also includes coverage information about scored coverages.

[Figure 7-49 on page 377](#) displays the *Coverage Top-level Summary Report* section.

Figure 7-49 Top-Level Summary Report Instance-Based



The above report prints Average grades as well as Covered grades of the reported metrics types. The covered grades include absolute coverage numbers that are shown as (Covered items / Total items / Marked items) for selected metrics types. For details on grade calculation, see [Coverage Grading in IMC](#) on page 407.

In the case of instance-based HTML report, the *name* column shows any of the following:

- The name of the top-level instance as a hyperlink (if the design has a single top-level instance). You can navigate through the instance hierarchy by clicking on the top-level instance name.
- Text [Instance Coverage](#) as a hyperlink (if the design has multiple top-level instances). You can view the list of top-level instances by clicking on the hyperlink [Instance Coverage](#) and then navigate through individual instance hierarchies.

Note: If the design has multiple top-level instances, the `index.html` page shows cumulative coverage of all the top-level instances in the *Overall Instance-Based Coverage* table.

To navigate through the design hierarchy:

1. Click the `dtrmf_recv_core` link.

Figure 7-50 on page 378 displays the coverage of instances within `dtrmf_recv_core`.

Figure 7-50 Instance-Based Report

Coverage Summary Report, Instance-Based												
Top Level Summary												
Instance name: <code>dtrmf_recv_core</code> (Type name: <code>dtrmf_recv_core</code>)												
Code Average	Code Covered	Block Average	Block Covered	Expression Average	Expression Covered	Toggle Average	Toggle Covered	Fsm Average	Fsm Covered	Functional Average	Functional Covered	name
64.64%	56.94% (2736/4805/17)	79.87%	49.31% (430/872/14)	45.61%	41.63% (92/221/3)	56.41%	59.64% (2214/3712)	59.11%	44.67% (109/244)	60.85%	37.86% (39/103)	Cumulative
Code	Code Covered	Block	Expression	Toggle	Fsm	Fsm Covered	Functional	Functional Covered	name			
53.10%	53.10% (154/290)	n/a	n/a	53.10% (154/290)	n/a	n/a	n/a	n/a	Self			
Coverage of immediate sub-instances:												
Code Average	Code Covered	Block Average	Block Covered	Expression Average	Expression Covered	Toggle Average	Toggle Covered	Fsm Average	Fsm Covered	Functional Average	Functional Covered	name
58.33%	58.33% (7/12)	n/a	n/a	n/a	n/a	58.33%	58.33% (7/12)	n/a	n/a	n/a	n/a	TEST_CONTROL_INST
90.00%	90.00% (9/10/3)	n/a	n/a	90.00% (0/0/3)	n/a	90.00%	90.00% (9/10)	n/a	n/a	n/a	n/a	ROM_512x16_INST
64.10%	57.63% (68/118/3)	100.00%	100.00% (3/3/3)	n/a	n/a	56.64%	56.52% (65/15)	n/a	n/a	n/a	n/a	RAM_128x16_TEST_INST
53.13%	38.97% (53/136/6)	100.00%	100.00% (3/3/3)	n/a	n/a	37.60%	37.59% (50/133)	n/a	n/a	n/a	n/a	RAM_256x16_TEST_INST
66.67%	66.67% (34/51)	66.67%	66.67% (4/6)	n/a	n/a	66.67%	66.67% (30/45)	n/a	n/a	n/a	n/a	TDSP_DS_CS_INST

The page displays the coverage for instances under `dtrmf_recv_core`. It also includes a link named `Self`, using which you can view the detailed report for the `dtrmf_recv_core` instance.

You can view the detailed report of any of the instances within `dtrmf_recv_core` by clicking its hyperlink in the `name` column.

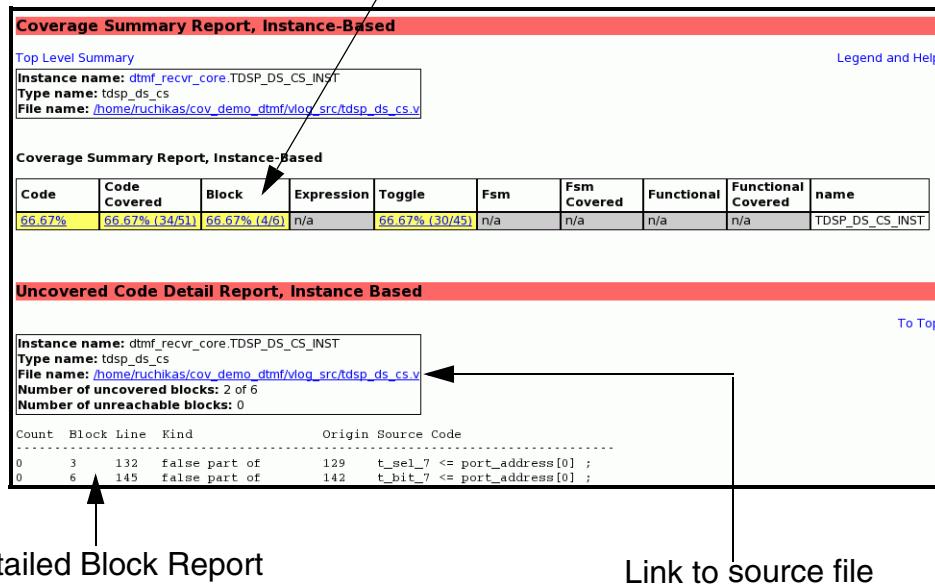
Note: You can also sort coverage data in different tables by clicking a column header.

2. To view detailed coverage for `TDSP_DS_CS_INST`, click its hyperlink in the `name` column.

Figure 7-51 on page 379 displays the detailed report of `TDSP_DS_CS_INST`.

Figure 7-51 Instance-Based Detailed Report

Local summary with coverage numbers as links to detailed reports



The above report displays the block coverage report of TDSP_DS_CS_INST. You can view the detailed report of other metrics types of this instance by clicking the hyperlinks in the report header.

Similarly, you can navigate through other instances of the design.

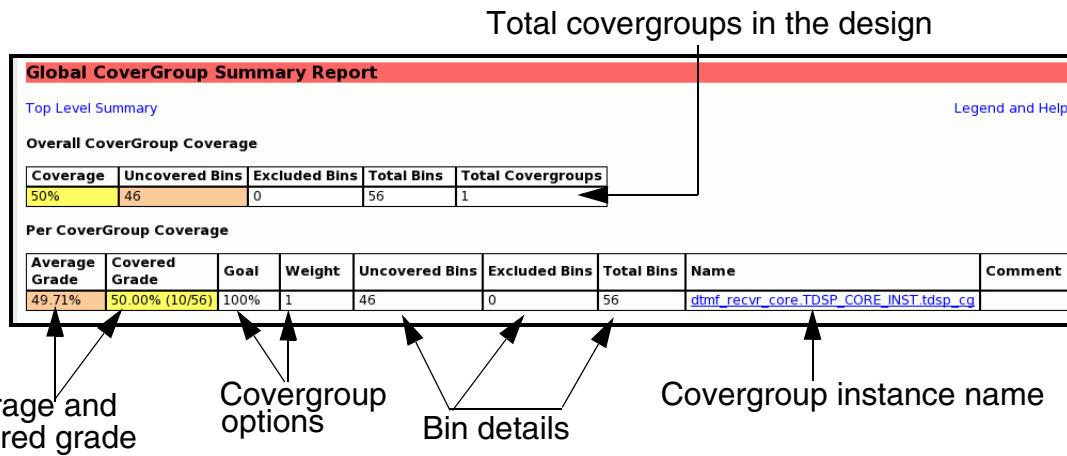
Important

You can move the HTML reports to another location for analysis. When you move HTML files, the links to source files break if the absolute source path is not available on the system where the HTML report is viewed. If the absolute source path is available, links to source files work fine. All other navigation links work fine even if the absolute source path is not available.

Global Covergroup Summary

Figure 7-52 on page 380 displays the *Global Covergroup Summary* section that is displayed when you click the *Global Covergroup Summary* link on the `index.html` page.

Figure 7-52 Global Covergroup Summary



The *Global Covergroup Summary* section provides a global view of all covergroups regardless of location. This section has two tables:

- Overall Covergroup Coverage, which summarizes coverage of all covergroups in the design.
- Per Covergroup Coverage, which lists coverage for each covergroup.

Overall Covergroup Coverage

The *Overall Covergroup Coverage* table displays the overall coverage of all of the covergroups, bin information, and total covergroups in the design. The bin information shown in the table is based on the options specified while generating the report.

- If `-covered` option is used, *Covered Bins* and *Total Bins* are shown.
- If `-uncovered` option is used, *Uncovered Bins* and *Total Bins* are shown.
- If `-all` option is used, *Uncovered Bins*, *Excluded Bins*, and *Total Bins* are shown.
- If `-excludes` option is used, *Excluded Bins* and *Total Bins* are shown.
- If `-both` option is used, *Covered Bins*, *Uncovered Bins*, and *Total Bins* are shown.

Note: If none of the options is specified, `-uncovered` is assumed.

The overall coverage shown in the *Coverage* column is calculated as:

Coverage of all the covergroups / Total covergroups in the design

Per Covergroup Coverage

The *Per Covergroup Coverage* table displays information about actual coverage, covergroup options, and bin details for different covergroups/covergroup instances in the design.

Note: The covergroup options control the behavior of the covergroup, coverpoint, and cross. For details on covergroup options, see the *ICC User Guide*.

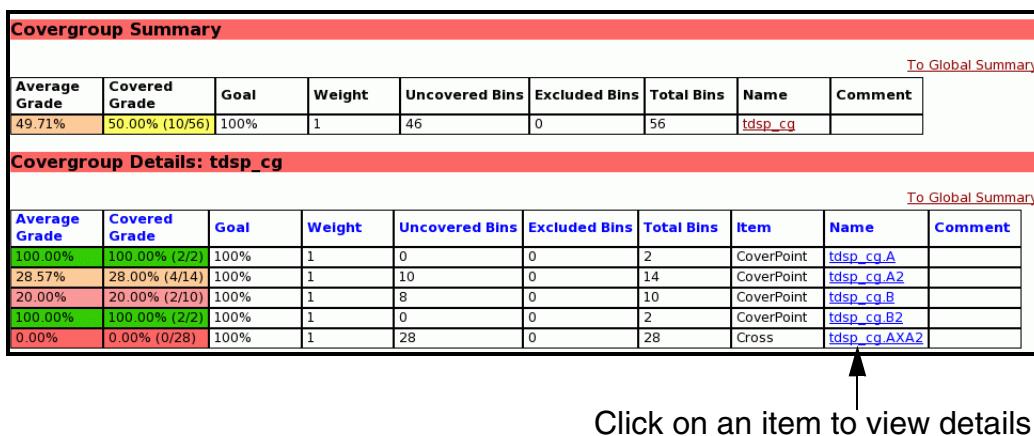
The *Per Covergroup Coverage* table also allows you to navigate through the covergroup items, such as coverpoints and crosses within the covergroup.

To navigate through the covergroup items:

1. Click the `dmtf_recv_core.TDSP_CORE_INST.tdsp_cg` link.

[Figure 7-53](#) on page 381 displays the details of covergroup instance `tdsp_cg`.

Figure 7-53 Details of `tdsp_cg`



Covergroup Summary									To Global Summary
Average Grade	Covered Grade	Goal	Weight	Uncovered Bins	Excluded Bins	Total Bins	Name	Comment	
49.71%	50.00% (10/56)	100%	1	46	0	56	tdsp_cg		
Covergroup Details: <code>tdsp_cg</code>									
Average Grade	Covered Grade	Goal	Weight	Uncovered Bins	Excluded Bins	Total Bins	Item	Name	Comment
100.00%	100.00% (2/2)	100%	1	0	0	2	CoverPoint	tdsp_cg.A	
28.57%	28.00% (4/14)	100%	1	10	0	14	CoverPoint	tdsp_cg.A2	
20.00%	20.00% (2/10)	100%	1	8	0	10	CoverPoint	tdsp_cg.B	
100.00%	100.00% (2/2)	100%	1	0	0	2	CoverPoint	tdsp_cg.B2	
0.00%	0.00% (0/28)	100%	1	28	0	28	Cross	tdsp_cg.AXA2	

Click on an item to view details

The page displays the covergroup summary and also detailed coverage information about coverpoints and crosses within the covergroup instance `tdsp_cg`. The *Coverage* column shows the actual coverage for each coverage item. For details on setting covergroup options (`goal` and `weight`), see the *ICC User Guide*.

2. To display the bins within a coverpoint or a cross, click its hyperlink in the *Name* column. Click the `tdsp_cg.A2` coverage point.

[Figure 7-54](#) on page 382 displays the details of covergroup `tdsp_cg.A2`.

Figure 7-54 Details of Coverpoint tdsp_cg.A2

Hit count for specific bins

Bins within coverpoint A2

CoverPoint Details: tdsp_cg.A2

To Global Summary | To CoverGroup

Average Grade	Covered Grade	Goal	Weight	Uncovered Bins	Excluded Bins	Total Bins	Item	Name	Comment
26.57%	26.00% (4/14)	100%	1	10	0	14	CoverPoint	A2	

Count **Atleast** **Bin Name**

42	40	vector_low[18]
45	40	vector_low[19]
42	40	vector_low[20]
42	40	vector_med[20]
36	40	vector_med[21]
36	40	vector_med[22]
15	40	vector_med[23]
0	40	vector_med[24]
6	40	vector_med[25]
0	40	vector_med[26]
0	40	vector_med[27]
0	40	vector_med[28]
0	40	vector_med[29]
0	40	vector_med[30]

Minimum hits for a bin to be considered covered

Similarly, you can navigate through other covergroup items.

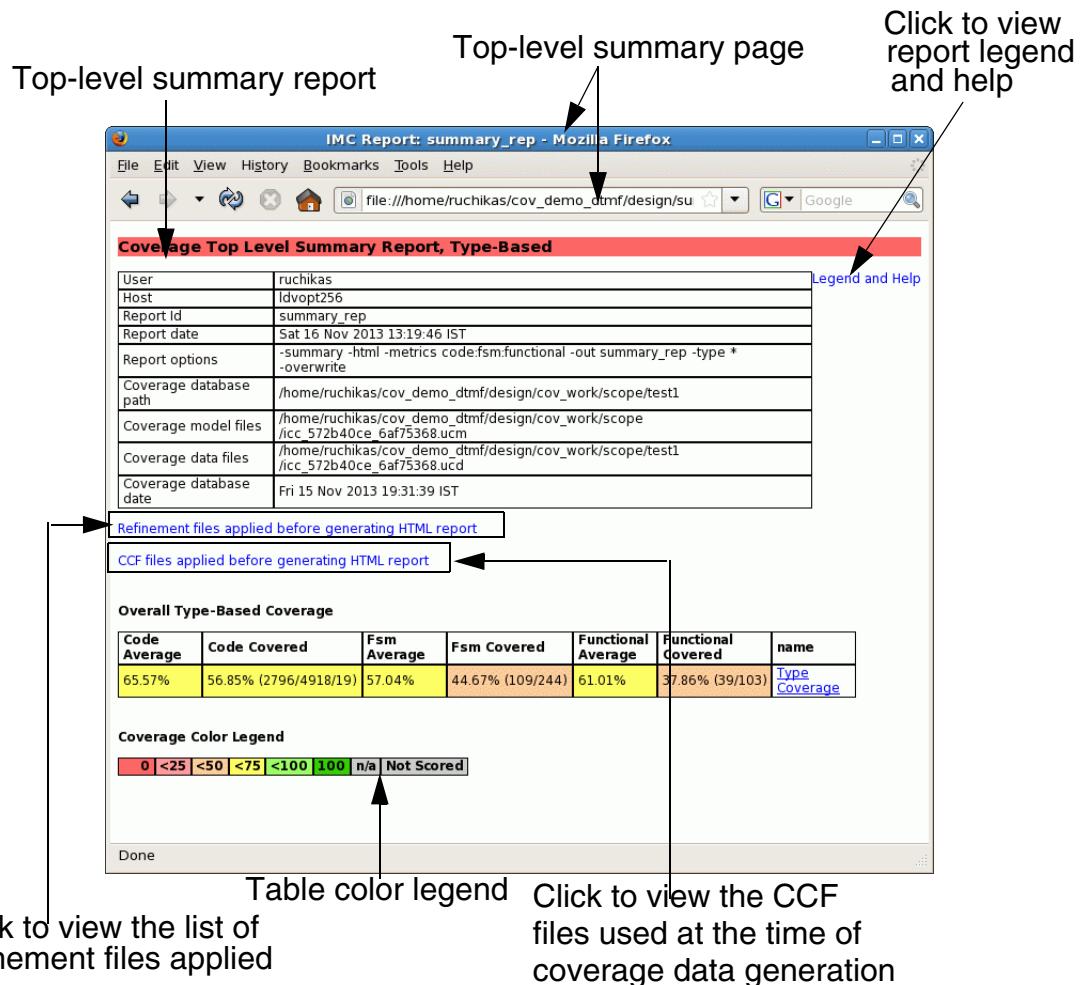
Consider another example of a type-based summary HTML report generated using:

```
report -summary -html -out summary_rep -overwrite -metrics code:fsm:functional
-type *
```

To view HTML reports generated with the above command, open `summary_rep/index.html` in a standard web browser.

[Figure 7-55](#) on page 383 shows the top-level summary page `index.html` in the browser.

Figure 7-55 Top-Level Summary Page



Similar to an instance-based report, it displays information such as who generated the report, when the report was generated, the options used to generate the report, coverage database location, model file (.ucm) and data file (.ucd) used to generate the report, and time when the coverage database was generated.

Note: A summary report does not provide a link to access the global covergroup summary. Global covergroup summary is available only with detailed reports.

To view the type-based *Coverage Summary Report*, click the [Type Coverage](#) hyperlink.

[Figure 7-56 on page 384 shows the *Coverage Summary Report*.](#)

Figure 7-56 Coverage Summary Report

Coverage Summary Report, Type-Based						
Top Level Summary				Legend and Help		
Overall Type-Based Coverage						
Code Average	Code Covered	Fsm Average	Fsm Covered	Functional Average	Functional Covered	name
65.57%	56.85% (2796/4918/19)	57.04%	44.67% (109/244)	61.01%	37.86% (39/103)	Type Coverage

Coverage Summary Report, Type-Based						
Code	Code Covered	Fsm	Fsm Covered	Functional	Functional Covered	name
53.10%	53.10% (154/290)	n/a	n/a	n/a	n/a	dtnf_recv_core
54.17%	54.55% (12/22)	n/a	n/a	n/a	n/a	test_control
90.00%	90.00% (9/10/5)	n/a	n/a	n/a	n/a	rom_512x16
46.55%	43.33% (26/60)	n/a	n/a	n/a	n/a	ram_128x16_test
85.09%	72.58% (45/62/3)	n/a	n/a	n/a	n/a	ram_128x16
43.66%	37.68% (26/69)	n/a	n/a	n/a	n/a	ram_256x16_test

You can sort coverage data in the table by clicking a column header.

Note: When using the `report` command, if `-summary` or `-detail` is not specified, then `-detail` is assumed in the case of HTML reports and `-summary` is assumed in the case of text reports.

Example: Cross Information in HTML Reports

The `-cross <expand> | <aggregate>` option of the `report` command specifies if the tuple information of automatically generated cross bins must be printed separately or after grouping similar cross bins.

By default, the similar cross bins are not grouped and the cross tuple/bin information is printed as a list, as shown in [Figure 7-57](#) on page 385.

Figure 7-57 Cross Bins Information Reported as a List (Default Behavior)

CoverPoint Details: cg_inst.cross_AXB									
To Global Summary To CoverGroup									
Average Grade	Covered Grade	Goal	Weight	Uncovered Bins	Excluded Bins	Total Bins	Item	Name	Comment
63%	62% (10/16)	100%	1	6	0	16	Cross	cross_AXB	
Count Atleast Bin Name									
25	1	<auto[0],auto[3]>							
1	1	<auto[1],auto[2]>							
1	1	<auto[1],auto[3]>							
1	1	<auto[2],auto[1]>							
1	1	<auto[2],auto[2]>							
1	1	<auto[2],auto[3]>							
1	1	<auto[3],auto[0]>							
1	1	<auto[3],auto[1]>							
1	1	<auto[3],auto[2]>							
1	1	<auto[3],auto[3]>							

Cross bins printed as a list

To group similar cross bins, use the `-cross aggregate` option, as shown below:

```
report -detail -all -html -out detail_rep2 -cross aggregate -inst *...
```

With the above command, the similar bins are grouped together to identify patterns. These patterns are then reported, as shown in [Figure 7-58](#) on page 386.

Figure 7-58 Similar Cross Bins Grouped to Identify Patterns (with -cross aggregate option)

Coverpoints participating in the cross

CoverPoint Details: cg_inst.cross_AXB

[To Global Summary](#) | [To CoverGroup](#)

Average Grade	Covered Grade	Goal	Weight	Uncovered Bins	Excluded Bins	Total Bins	Item	Name	Comment
63%	62% (10/16)	100%	1	6	0	16	Cross	cross_AXB	

Notes:
- Coverbins are grouped together to identify patterns.
- Groups can be overlapping.

Uncovered:

A	B	Size	Atleast	Count
auto[0], auto[1], auto[2]	auto[0]	3	1	0
auto[0]	auto[0], auto[1], auto[2]	3	1	0
auto[0], auto[1]	auto[0], auto[1]	4	1	0

Covered:

A	B	Size	Atleast	Count
*	auto[3]	4	1	28
auto[1], auto[2], auto[3]	auto[2], auto[3]	6	1	6
auto[2], auto[3]	auto[1], auto[2], auto[3]	6	1	6
auto[3]	*	4	1	4

Excluded:
No items

Illegal/Ignore:
No items

Similar cross bins grouped together to identify patterns

Number of bins grouped together

In the above report,

- Each row indicates the group of similar cross bins.
- Columns *A* and *B* indicate the coverpoints participating in the cross.

Note: These columns might sometimes show a * to indicate all values of that coverpoint. When you place the cursor over *, bins of that coverpoint (that are associated with *) are shown.

- The *Size* shows the number of cross bins grouped together.

- The *Atleast* column shows the value of the `atleast` covergroup option.

For details on covergroup options, see the *ICC User Guide*.

- The *Count* column shows the number of times that group was hit.

When using the `-cross` aggregate option, remember that:

- The `cross_num_print_missing` covergroup option is not considered and is ignored while grouping similar bins.
- Grouping of cross bins is applicable only to automatically generated cross bins.

Note: The `-cross` option is supported only with HTML detail reports.

Example: Bin Information in HTML Reports

The `-kind <expand> | <compact>` option of the `report` command specifies the format in which the covergroup bin information must be printed in the HTML report.

By default, bins are reported as a list without grouping similar bins, as shown in [Figure 7-59](#) on page 387.

Figure 7-59 Bin Information Reported as a List (Default Behavior)

CoverPoint Details: c1::cg.A									
To Global Summary To CoverGroup									
Average Grade	Covered Grade	Goal	Weight	Uncovered Bins	Excluded Bins	Total Bins	Item	Name	Comment
43%	43% (6/14)	100%	1	8	0	14	CoverPoint	A	
Bins printed as a list									
Count	Atleast	Bin Name							
0	1	Ab1[0]							
2	1	Ab1[1]							
3	1	Ab1[2]							
0	1	Ab1[3]							
1	1	Ab1[4]							
0	1	Ab1[5]							
0	1	Ab1[6]							
1	1	Ab1[7]							
2	1	Ab1[8]							
1	1	Ab1[9]							
0	1	Ab1[10]							
0	1	Ab1[11]							
0	1	Ab1[12]							
0	1	Ab1[13]							

In the above report, the bin information is printed as a list in a single table.

Note: Only 5000 bins can be displayed in a single table. If the number of bins is more than 5000, then a new table is created to show the next set of bins. The new table is shown on a separate page. In addition, sorting applied on a previous page will not be available on the next page.

In this format, coverage analysis becomes difficult and time-consuming (if there are a large number of bins).

Using the `-kind compact` option of the `report` command, you can print the bin information in an aggregated format. In this format, similar bins are grouped together (to create cubes), which enables you to quickly analyze bins. This format might increase computational overhead; however, it makes data analysis more efficient and less time-consuming.

To group similar bins, use the `-kind compact` option, as shown below:

```
report -detail -html -kind compact -out html_compact
```

With the above command, the similar bins are grouped together to identify patterns. These patterns are then reported, as shown in [Figure 7-60](#) on page 388.

Figure 7-60 Similar Bins Grouped Together (with `-kind compact` option)

CoverPoint Details: c1::cg.A										
To Global Summary To CoverGroup										
Average Grade	Covered Grade	Goal	Weight	Uncovered Bins	Excluded Bins	Total Bins	Item	Name	Comment	
43%	43% (6/14)	100%	1	8	0	14	CoverPoint	A		
Uncovered:										
Size	Count	A								
1	0	Ab1[0]								
1	0	Ab1[3]								
2	0	Ab1[5]..Ab1[6]								
4	0	Ab1[10]..Ab1[13]								
Covered:										
Size	Count	A								
2	5	Ab1[1]..Ab1[2]								
1	1	Ab1[4]								
3	4	Ab1[7]..Ab1[9]								

Number of bins grouped together

Similar bins grouped together to identify patterns

In the above report:

- Separate tables are created for uncovered and covered bins.

- Each row indicates the group of similar bins. You can click any of the rows to view details of bins in that group.
- The *Size* column shows the number of bins grouped together.
- The *Count* column shows the number of times that group was hit.
- Columns *A* indicates the name of the coverpoint and lists the bins associated with that coverpoint. When you place the mouse over the values shown in this column, the bins of that coverpoint are shown as a tooltip. In case the tooltip is too long, use the mouse wheel to scroll down and view all the values.

To view the details of bins in a particular group, click the group or click anywhere in the row corresponding to that group.

[Figure 7-61](#) on page 389 shows details of group Ab1[10] . . . Ab1[13].

Figure 7-61 Bins in the Selected Group

Count	Atleast	A
0	1	Ab1[10]
0	1	Ab1[11]
0	1	Ab1[12]
0	1	Ab1[13]

The above report shows the list of bins in group Ab1[10] . . . Ab1[13]. In the report, the *Atleast* column shows the value of the *atleast* covergroup option. For details on covergroup options, see the *ICC User Guide*.

Note: You can turn off the generation of the detailed information page shown in [Figure 7-61](#) on page 389 by using the `-cubeExpand off` option with the `-kind compact` option at the time of generating the report. Turning off this option is useful when the information on the detailed page is not of much relevance and also in situations where you want to save the disk space.

Consider another example shown in [Figure 7-62](#) on page 390. This report shows how cross bins are reported when the `-kind compact` option is used.

Figure 7-62 Similar Cross Bins Grouped Together (with -kind compact option)

Coverpoints participating in the cross

CoverPoint Details: c1::cg_ A_X_B_0

[To Global Summary](#) | [To CoverGroup](#)

Average Grade	Covered Grade	Goal	Weight	Uncovered Bins	Excluded Bins	Total Bins	Item	Name	Comment
11%	11% (9/84)	100%	1	75	0	84	Cross	A_X_B_0	

Uncovered:

Size	Count	A	B
6	0	Ab1[0]	*
4	0	Ab1[1]	Bb1[0]..Bb1[1], Bb1[4]..Bb1[5]
4	0	Ab1[2]	Bb1[0]..Bb1[1], Bb1[3]..Bb1[4]
6	0	Ab1[3]	*
5	0	Ab1[4]	Bb1[0]..Bb1[3], Bb1[5]
12	0	Ab1[5]..Ab1[6]	*
5	0	Ab1[7]	Bb1[0], Bb1[2]..Bb1[5]
4	0	Ab1[8]	Bb1[1]..Bb1[4]
5	0	Ab1[9]	Bb1[0], Bb1[2]..Bb1[5]
24	0	Ab1[10]..Ab1[13]	*

Covered:

Size	Count	A	B
2	2	Ab1[1]	Bb1[2]..Bb1[3]
2	3	Ab1[2]	Bb1[2], Bb1[5]
1	1	Ab1[4]	Bb1[4]
1	1	Ab1[7]	Bb1[1]
2	2	Ab1[8]	Bb1[0], Bb1[5]
1	1	Ab1[9]	Bb1[1]

Number of bins grouped together

Similar bins grouped together to identify patterns

In the above report:

- Separate tables are created for uncovered and covered cross bins.
- Each row indicates the group of similar cross bins. You can click any of the rows to view details of bins in that group.
- The *Size* column shows the number of cross bins grouped together.
- The *Count* column shows the number of times that group was hit.
- Columns *A* and *B* indicate the cross dimensions (coverpoints participating in the cross). When you place the mouse over the column values, the bins of that coverpoint are shown as a tooltip.

Note: These columns might sometimes show a * to indicate all values of that coverpoint. When you place the cursor over *, bins of that coverpoint (that are associated with *) are shown.

Important points related to -kind compact option

When using the -kind compact option, remember that:

- In the case of large items (with more than 1 million bins), a compact report without expansion is generated.
- Grouping of bins is applicable only to automatically generated bins.
- Grouping of bins does not apply to following items:
 - User-defined bins
 - Ungradable bins
 - SystemVerilog transition bins
 - Overlapping bins

Note: In such cases, by default, the compact option is automatically turned OFF and the bin information is printed as a list (as with the -kind expand option).

7.2.2 The report_metrics Command

The report_metrics command generates metrics report in an HTML format. The report generated using the report_metrics command includes hyperlinks to navigate through the design hierarchy and is also aligned with the reports generated from IMC GUI.

The BNF of the report_metrics command is:

```
report_metrics
[-out <output_directory>]
[-overwrite]
[-title <report_title>]
[-view <view> [-filter <filter>]]
[-report_type regular | tabulated | extended]
[-extended true | false]
{[-summary] | [-detail [-kind expand | aggregate] [-metrics <metrics_type>]
[-source on|off]
[-exclComments]
[-aspect sim | formal | both]
[-assertionStatus]
[-allAssertionCounters]
[-covered | -uncovered | -both | -all | -excludes]}}
[-type |-inst]
[list]
```

```
metrics_type ::= [all] [code] [fsm] [functional] [block] [expression] [toggle]  
[state] [transition] [arc] [covergroup] [assertion]
```

where

- `-out <output_directory>` redirects the report files to an alternate location instead of the default `html_<timestamp>` directory.

In the absence of the `-out` option, by default, a directory named `html_<timestamp>` is created in the current working directory, and coverage reports along with the top-level summary page (`index.html`) are stored in the `html_<timestamp>` directory.

Note: A top-level summary page `index.html` is used to navigate through the HTML reports available in the `html_<timestamp>` directory.

With the `-out` option, the command redirects the HTML output to `<output_directory>`. If `<output_directory>` includes just the name of the output directory, then the command creates the output directory named `<output_directory>` in the current working directory, and stores the HTML report files along with the top-level summary page in `<output_directory>`. For example, if `<output_directory>` is specified as:

```
report_metrics -out day1
```

then directory named `day1` is created in the current working directory to store HTML report files.

Note: If `<output_directory>` already exists, the output directory is not overwritten. You must use the `-overwrite` option to overwrite an existing output directory.

If `<output_directory>` includes the complete path, then the command creates a directory by the name mentioned at the lowest level in the path, and stores the report files at that location. For example, if `<output_directory>` is specified as:

```
report_metrics -out data/myreports/day1
```

then directory named `day1` is created under `data/myreports` to store report files. In this case, the path (`data/myreports`) must exist. The directory `day1` must not exist, as it will be created and not overwritten unless the `-overwrite` option is used.

- `-overwrite` enables overwriting of the existing output directory. By default, the output directory is not overwritten and an error is reported.
- `-title <report_title>` allows you to specify a title for the report. The report title cannot include blank spaces. In the absence of this option, by default, the title is specified as *Metrics Report*.
- `-view <view_name>` allows you to generate a report using a specific view that would have been created using IMC GUI. In the absence of this option, the default view, which

is *All_Metrics* is used to generate the report. For details on how to create a view in IMC GUI, see [Defining and Organizing Views](#) on page 72.

For example, if a view named `Mike` is already defined and you want to generate a report using the view `Mike`, use the following command:

```
report_metrics -view Mike
```

- ❑ `-filter <filter>` specifies filter on the specified view. The format for specifying the filter is:

```
[attribute] : [operator] [value]
```

where operator can be any of the: {`>`, `<`, `=`, `<=`, `>=`, `=`, `!=`, `range`, `~`, `!~`}

For example:

```
-filter name:~comp
```

You can also specify multiple filters on the command line. For example:

```
-filter owner:~shaim -filter status:=passed
```

Note: The filter specified on the command line will override the existing value in the view. In addition, an error is reported if the `attribute_name` specified with the filter does not exist.

- `-report_type` allows you to specify the type of bin-level report to be generated. It can be any of the following:

- ❑ Regular: This is the default report. It lists nodes and sub-nodes and provides hyperlinks for navigating further. To generate a regular report, use:

```
report_metrics -report_type regular
```

- ❑ Tabulated: This generates a tabulated bin-level report. This report shows the *Metrics* tree and bin details in tables side-by-side. For more details, see [Example: Bin-Level Tabulated Report](#) on page 325. To generate a bin-level tabulated report, use:

```
report_metrics -report_type tabulated
```

- ❑ Extended: This generates an extended report. This report contains a single table in which the *Metrics* tree and the bins information for each of the cover items is shown in the tree. For more details, see [Example: Bin-Level Extended Report](#) on page 326. To generate an extended report, use:

```
report_metrics -report_type extended
```

Note: In the absence of `-report_type` option, a regular report is generated.

- `-extended true | false` enables or disables extended report that is — showing extended tree (list covergroups, cover items, FSMs, and assertions along with the types and instances) in the hierarchy tree. By default, this option is disabled. If not specified,

the report will be extended according to the configuration option *Show extend metrics tree*.

- `-summary | -detail` specifies the type of regular report that must be generated. If not specified, `-summary` is assumed. With the `-detail` option, you can specify the metrics types for which the report must be generated. The metrics types can be specified using the `-metrics <metrics_type>` option. The `<metrics_type>` can be any, or a combination of the following:
 - `code` for printing a detailed block, expression, and toggle coverage report
 - `fsm` for printing a detailed state, transition, and arc coverage report
 - `functional` for printing a detailed assertion and covergroup coverage report
 - `block` for printing a detailed block coverage report
 - `expression` for printing a detailed expression coverage report
 - `toggle` for printing a detailed toggle coverage report
 - `state` for printing a detailed state coverage report
 - `transition` for printing a detailed transition coverage report
 - `arc` for printing a detailed arc coverage report
 - `assertion` for printing a detailed assertion coverage report
 - `covergroup` for printing a detailed covergroup coverage report
 - `overall` for printing a detailed report for all metrics
 - `all` for printing a detailed report for all metrics

Note: You can specify more than one metrics type by separating the metric types with a colon (:). If `<metrics_type>` is not specified, a detailed report for all metrics is generated.

For example, to generate a summary report, use:

```
report_metrics -summary
```

or

```
report_metrics
```

For example, to generate a detailed report, use:

```
report_metrics -detail
```

To generate a detailed block report, use:

```
report_metrics -detail -metrics block
```

To generate a detailed block and toggle report, use:

```
report_metrics -detail -metrics block:toggle
```

- `-kind` specifies the format in which the covergroup bin information must be printed in the HTML report. The format can be specified as any of the following:
 - `expand`—to print the bin information as a list without grouping similar bins. This is the default behavior.
 - `aggregate`—to print the cross tuple information in a tabular format after grouping similar bins.
- `-source off | on` enables or disables reporting of the source text and line numbers in the detailed report. By default, this information is printed in the report. To disable printing of source text and line numbers, set the `-source` option to `off`.

Note: The `-source` option is relevant only with the `-detail` option. In addition, currently, the batch mode of IMC cannot read the compressed files, and therefore, the *Source Code* column of the report of entities from the compressed files will be blank (even if the `-source` option is turned on).

- `-exclComments` enables reporting of exclusion comment in the detailed report page of the HTML report. By default, this information is not printed in the report. When you use this option in the `report_metrics` command, an additional column, *Exclusion User, Reviewer, Comment* is printed in the detailed report page, which shows the exclusion comment information as:

```
[<user>-<reviewer>] : <comment>
```

where

- `<user>` is the login ID of the user.
- `<reviewer>` is the name of the exclusion reviewer. The exclusion reviewer is specified using the `-reviewer` option of the `exclude` command. If no reviewer is specified at the time of exclusion, the default value `unknown` is considered as the reviewer. For more details on the `exclude` command, see [Refinement in Command-Line Interactive Mode](#) on page 188.
- `<comment>` is the exclusion comment. The exclusion comment is specified using the `-comment` option of the `exclude` command. For more details on the `exclude` command, see [Refinement in Command-Line Interactive Mode](#) on page 188.

For example, if the `exclude` command was specified as:

```
exclude -inst dtmf_recv_core.SPI_INST -block 8 -comment "This block cannot be covered" -reviewer Ben
```

and the `report_metrics` command was specified as:

```
report_metrics -out detail_repl -detail -all -inst *... -exclComments
```

then an additional column `Exclusion User, Reviewer, Comment` with the following value will be printed in the detailed report page:

`[bensky-Ben]` : This block cannot be covered

where `bensky` is the login ID, `Ben` is the exclusion reviewer, and `This block cannot be covered` is the exclusion comment.

If the exclusion reviewer was not specified in the `exclude` command, then the `Exclusion User, Reviewer, Comment` column will show `[bensky-unknown]` : `This block cannot be covered`.

Note: The `-exclComments` option enables printing of exclusion comments specified at the time of exclusion. In case exclusion comments are not specified at the time of exclusion and only the exclusion reviewer is specified, then the `Exclusion User, Reviewer, Comment` column will not show anything even if the `-exclComments` option is used.

- `-aspect sim | formal | both` specifies the assertion properties to be shown in the report.
 - `sim` shows only simulation assertion properties in the report.
 - `formal` shows only formal assertion properties in the report.
 - `both` shows both simulation and formal assertion properties in the report.
- **Note:** By default, the value of the `-aspect` option is set to `sim`.
- `-assertionStatus` enables reporting of assertion status information in the detailed report. When you use this option, additional column(s) is shown in the report depending on the aspect specified with the `-aspect` option. The report shows following additional columns:
 - **Status** (if `-aspect` is `sim`). The Status can be any of the following:
 - `Passed` if the assertion finished without failure at least once, and never failed.
 - `Failed` if the assertion finished with failure at least once.
 - `Other` if the assertion never finished.
 - **Formal Status** (if `-aspect` is `formal`). The Formal Status can be any of the following:
 - `Proved` if the assertion finished without failure at least once, and never failed.
 - `Failed` if the assertion finished with failure at least once.
 - `Other` if the assertion never finished.

Note: If `-aspect` is specified as `both`, then both *Status* and *Formal Status* attributes are printed in the report.

Note: If `-allAssertionCounters` option is also used with the `-assertionStatus` option, then the value `Other` in the *Status* field is further classified as any of the following:

- `Vacuous` if the assertion was vacuous pass at least once but never finished.
- `Disabled` if the assertion transitioned to the disabled state at least once but never finished.
- `Attempted` if the assertion was attempted at least once but never finished.
- `Not Attempted` if the assertion was not attempted all (that is all the counters are equal to zero).

Note: The `-assertionStatus` option is applicable only for assertion coverage reports.

- `-allAssertionCounters` enables reporting of additional columns in the detailed report.
 - In case `-aspect` is specified as `sim`, then `Vacuous`, `Attempt`, and `Disabled` counters are also reported in addition to the default `Finished` and `Failed` counters.
 - In case `-aspect` is specified as `formal`, then `Formal Proved`, `Formal Passed`, and `Formal CEX` counters are also reported in addition to the default `Formal Finished` and `Formal Failed` counters.

Note: In case `-aspect` is specified as `both`, then counters reported with both (`sim` and `formal`) are printed in the report.

Note: The `-allAssertionCounters` option is applicable only if the `-abvrecordvacuous` switch is used at the time of simulation run. In case the `-abvrecordvacuous` switch is not used at simulation, then the `Vacuous`, `Attempt`, and `Disabled` columns will show `n/a`. For more details on the `-abvrecordvacuous` switch, see the *Verilog Compilation Command-Line Options* user guide.

- `-all` | `-covered` | `-uncovered` | `-both` | `-excludes` specifies whether to list instances/types for all, covered, uncovered, both covered and uncovered, or excluded items.

Note: If none of the options (`-all`, `-covered`, `-uncovered`, `-both`, or `-excludes`) is specified, `-uncovered` is assumed. In addition, these options are relevant only with the `-detail` option.

- `-type | -inst` specifies if report should be generated only for the types hierarchy or only for the instances hierarchy. If none of these options is specified, the report is generated for the instance hierarchy.
- `list` specifies the names of types or instances for which the report will be generated. The default value is `*...`, which matches all the top-level instances and their children. It can include wildcard characters `*` and `?`. Use `...` to specify all descendants of an instance. Instance names can be specified as hierarchical names, for example, `top.inst1.inst2`.

Note: The path can be prefixed with "*Verification\ Metrics*", "*Verification\ Metrics/Instances*", "*Verification\ Metrics/Types*", "*Instances*", or "*Types*". Prefix is not mandatory. However, if the *Instances* or *Types* prefix is specified, the performance of the command can be improved. For example, `report_metrics a.b.c` can be specified as:

```
report_metrics Verification\ Metrics/Instances/a.b.c  
or  
report_metrics Instances/a.b.c
```

Note: Wild cards (`*`, `?`, and `...`) are not supported in the prefix *Verification\ Metrics/Types* or *Verification\ Metrics/Instances*.

Examples

To generate detailed report only for type `spi`, use:

```
report_metrics -detail -type spi
```

To generate detailed report for all instances, use:

```
report_metrics -detail -inst *...
```

Example: Generating Instance-Based Detailed Report

Consider an example of an instance-based detailed report generated using:

```
report_metrics -out detail_rep1 -title My_Instance_Based_Report -detail -all -inst *...
```

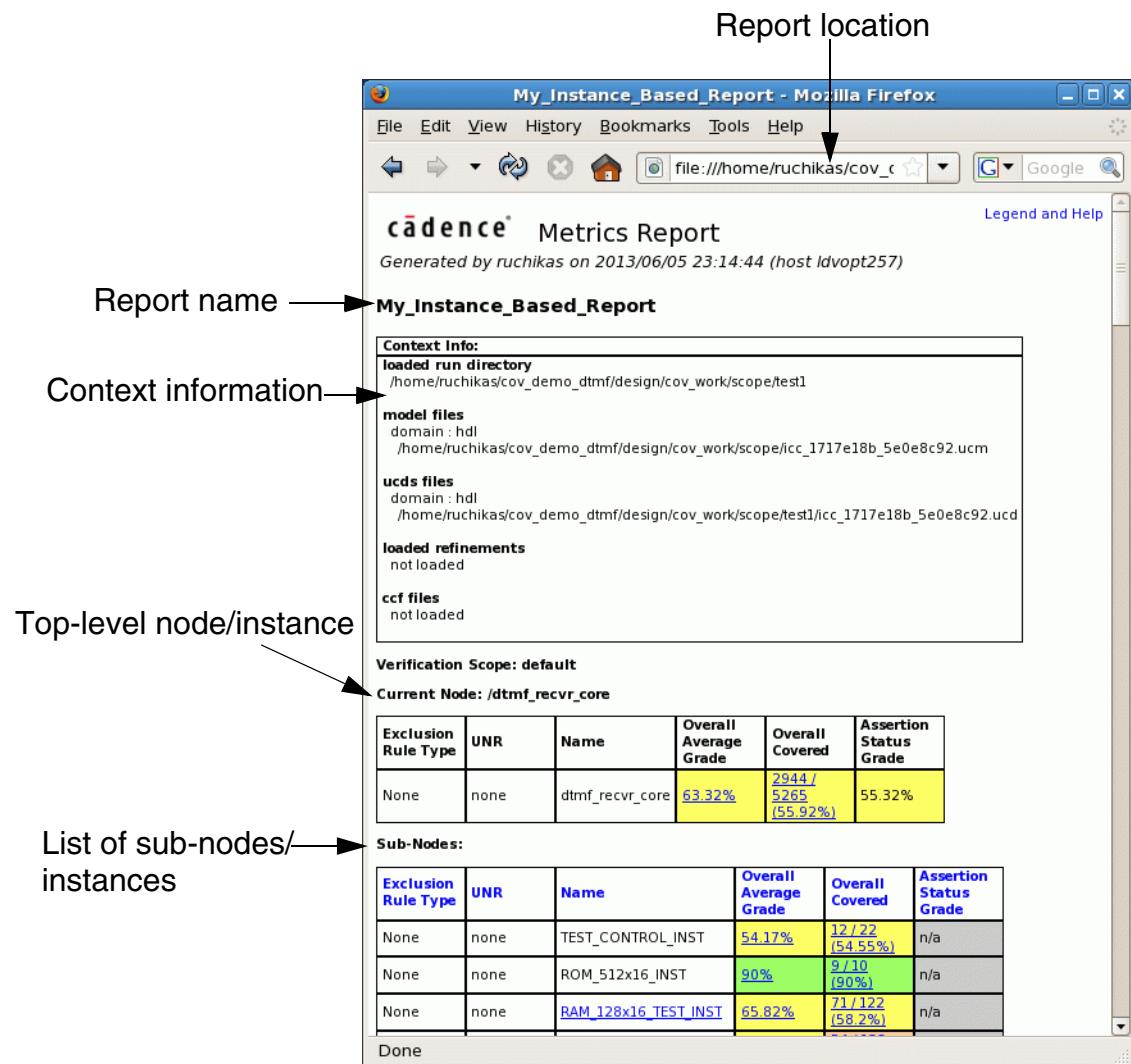
Note: The above command generates a detailed report for all metric types and for all instances. With the above command, the report is generated at:

```
<current_working_directory>/detail_rep1/index.html
```

To view the report, open `detail_rep1/index.html` in a standard web browser.

Figure 7-63 on page 399 shows the top-level page `index.html` in the browser.

Figure 7-63 Top-Level Page



The top-level page displays information such as, who generated the report and when the report was generated. It also shows the context information, such as coverage database location, model file (.ucm) and data file (.ucd) used to generate the report, the location and contents of the coverage configuration file, and the list of refinement files applied before generating the report.

The other details are shown based on the options specified at the time of generating the report. For example, in this case, all the instances are listed in a tabular format because instance-based report was generated. You can sort data in the table by clicking a column header.

The report output and the columns shown in the report are based on the view selected at the time of generating the report.

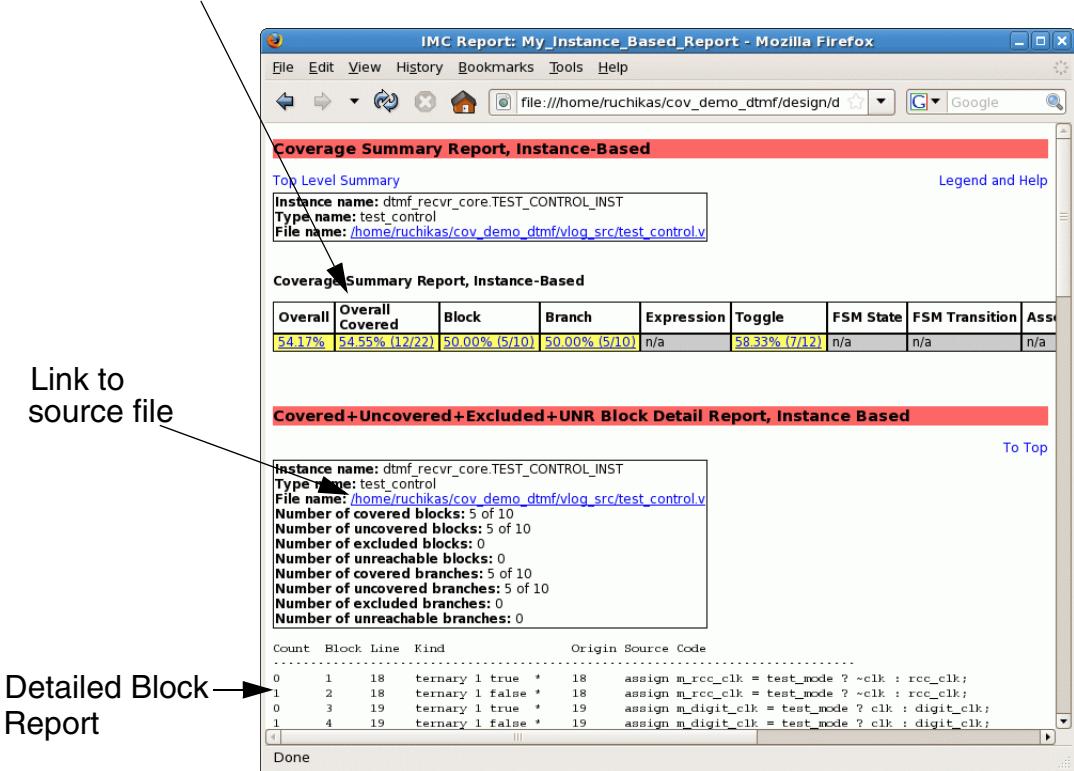
You can navigate through the report using the hyperlinks available in the report.

For example, in this report, you can view the detailed report of any of the instances by clicking the hyperlink in the *Overall Average Grade* column or the *Overall Covered* column. For example, to view a detailed report of instance TEST_CONTROL_INST, click the hyperlink in the *Overall Average Grade* column or the *Overall Covered* column.

[Figure 7-64 on page 400](#) displays the detailed report of TEST_CONTROL_INST.

Figure 7-64 Detailed Report

Summary Report showing coverage numbers as links to detailed reports



The above report displays the block coverage report of TEST_CONTROL_INST. You can view the detailed report of other metrics types of instance TEST_CONTROL_INST by clicking the hyperlinks in the *Coverage Summary Report, Instance-Based* table, shown in the above report.

Similarly, you can navigate through other instances of the report.

7.2.3 The report_unique Command

The `report_unique` command compares the entities covered in two sets of runs or just two runs and reports the unique coverage items and their contributing runs.

A unique coverage item is a leaf level coverage entity that is uniquely sampled in one of the sets.

Note: A coverage item can be considered unique in run A compared to run B, regardless of its existence in the model-definition of B.

The syntax of the `report_unique` command is:

```
report_unique
  {run_specification1} {run_specification2}
  [-out <out_path>]
  [-uniq_run_count <count>]
  [-metrics <metrics_type>]
  [-counts [-sort high_count | low_count]]
  [element_option]
  [-togglesContribByGrade]

run_specification1 ::= {-run1 <runpath> | -runfile1 <runfile>}
run_specification2 ::= {-run2 <runpath> | -runfile2 <runfile>}
<metrics_types> ::= [all] [code] [fsm] [functional] [block] [expression] [toggle]
[covergroup] [assertion]
element_option ::= -type <type> | -inst <instance> |
                  -entity <path> [-entity_type inst | type]
```

where

- `run_specification1` specifies the list of runs in the first group. You can specify the run list in any of the following ways:
 - At the command line using the `-run1 <runpath>` option.
 - In a text file using the `-runfile1 <runfile>` option. The `<runfile>` is the file that includes the list of runs to be included in the group. The runs in `<runfile>` should be listed one per line. You can include comments in `<runfile>` using #.

The runs specified in `<runpath>` or `<runfile>` can include just the run name or the complete path, as:

```
[[<workdir>/]<scope>/]<run>
```

- `run_specification2` specifies the list of runs in the second group. You can specify the run list in any of the following ways:

- At the command line using the `-run2 <runpath>` option.
- In a text file using the `-runfile2 <runfile>` option. The `<runfile>` is the file that includes the list of runs to be included in the group. The runs in `<runfile>` should be listed one per line. You can include comments in `<runfile>` using `#`.

The runs specified in `<runpath>` or `<runfile>` can include just the run name or the complete path, as:

```
[[<workdir>/]<scope>/]<run>
```

- `-out <out_path>` specifies the location where the result of the `report_unique` command must be stored. If not specified, the output of the command is redirected to the standard output device.
- `-metrics <metrics_type>` lets you specify the type of coverage that must be analyzed. The `<metrics_type>` can be any, or a combination, of the following:
 - `all` for all currently supported metrics.
 - `code` for block and toggle coverage
 - `functional` for both assertion and covergroup coverage
 - `block` for block coverage
 - `toggle` for toggle coverage
 - `assertion` for assertion coverage
 - `covergroup` for covergroup coverage

By default, all supported metric types are reported.

Note: You can specify more than one metric type by separating the metric types with a colon (`:`). For example, to specify block and expression, use `block:expression`. Currently, unique coverage report is not supported for expression and FSM metrics types.

- `-uniq_run_count <count>` specifies the maximum number of contributing runs that must be listed in the report for each unique item. If a particular unique item is sampled more than the value specified in `<count>`, then the first `<count>` runs are reported. Default value of this option is 1. In the absence of this option, one run is reported for each unique entity. For example, consider the following output:

```
"unique_entities": [  
  {  
    "type": "coverbin",  
    "index": 0,  
    "full_path": "sys/ascending/x/4",  
    "runs": [
```

```
        {"path": "cov_work/scope/count_by_seed_sn1"},  
        {"path": "cov_work/scope/count_by_seed_sn2"},  
        {"path": "cov_work/scope/count_by_seed_sn3"}  
    ],  
    "hitting_runs": 5  
,  
...  
}
```

In this case, 3 runs are listed for unique item `sys/ascending/x/4` because the `-uniq_run_count` was set as 3.

Note: A high value for `-uniq_run_count` might impact the performance of the operation.

- `-counts` is used to report the hit count along with each of the listed run. By default, the counters are not reported/collected. The use of this option might significantly impact the performance of the operation. For example, consider the following output:

```
"unique_entities": [  
    {  
        "type": "coverbin",  
        "index": 0,  
        "full_path": "sys/ascending/x/4",  
        "runs": [  
            {"path": "cov_work/scope/count_by_seed_sn1", "count": 1},  
            {"path": "cov_work/scope/count_by_seed_sn2", "count": 2},  
            {"path": "cov_work/scope/count_by_seed_sn3", "count": 3}  
        ],  
        "hitting_runs": 5  
    },  
...  
]
```

When you use the `-counts` option, for each of the listed run, the hit count also appears to indicate that the listed run was hit # times. In the absence of this option, the count mentioned in the above output snippet is not reported.

- `-sort` determines the sorting order of the listed runs per each unique bucket according to hit count. This option is applicable only when `-counts` option is used. The possible values with `-sort` are:

- ❑ `none` means no sort order is applied. It is same as `unsorted`.
- ❑ `low_count` sorts the runs in ascending order based on the hit count.
- ❑ `high_count` sorts the runs in descending order based on the hit count.

- ❑ `unsorted` will list the runs according to their order in the runfile (unsorted by the hit count).

Note: Default value is `unsorted`.

For example, consider the following output:

```
"unique_entities": [
  ...
  "full_path": "sys/ascending/*/4",
  "runs": [
    {"path": "cov_work/scope/count_by_seed_sn1", "count": 1},
    {"path": "cov_work/scope/count_by_seed_sn2", "count": 2},
    {"path": "cov_work/scope/count_by_seed_sn3", "count": 3}
  ...
]
```

In this case, the sort order is `low_count` because the runs are listed in the ascending order.

- `element_option` specifies the entities that need to be analyzed for `report_unique`.

By default, the `report_unique` is done for the complete model. To consider only specific entities during `report_unique` operation, use the `element_option` as:

- ❑ `-type <types>` to specify the modules for which unique entities should be reported.
- ❑ `-inst <instances>` to specify the instances for which unique entities should be reported.
- ❑ `-entity <path>` to specify instance or type for which unique entities should be reported. The `-entity_type` option can be specified with the `-entity` option. Valid values for entity type are: `type` and `inst`.

Wildcard characters are supported in specifying the entities.

- `-togglesContribByGrade`—When this option is set, only unique toggle transitions which contribute to the grade, will be reported. This means that—if for example, `sig1.rise` is unique in one of the run groups, but `sig1.fall` is not hit in any of the runs in both of the groups, it will not be reported. If this option is not set, all unique toggle transitions will be reported.

Examples:

To print a report that compares covered entities in a merged run cov_work/scope/merged and group of runs which are listed in a file named runsspec, use:

```
imc> report_unique -run1 merged -runfile2 runsspec
```

To redirect the output of report_unique command to a file named unique_entities.rpt, and to list up to three covering runs for each unique entities, use:

```
imc> report_unique -runfile1 runsspec1 -runfile2 runsspec2 -out unique_entities.rpt -uniq_run_count 3
```

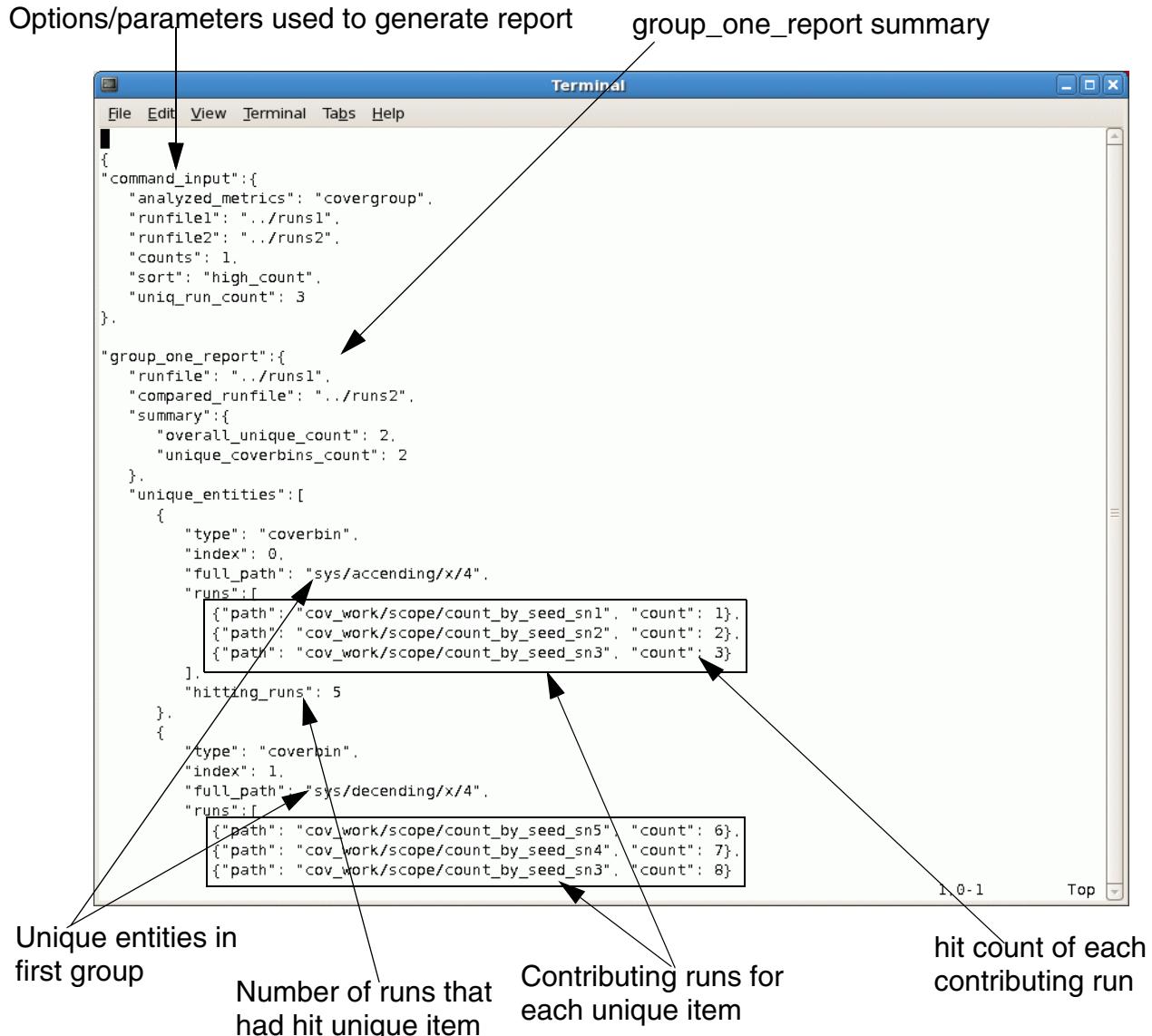
To list up to four covering runs for each unique entities and to show the hit count for each of the covering runs and to also sort the runs in ascending order, use:

```
imc> report_unique -runfile1 runsspec1 -runfile2 runsspec2 -uniq_run_count 3 -counts -sort low_count
```

[Figure 7-65 on page 406](#) shows sample output of the report_unique command.

Note: Unlike other commands, the output of this command is in a JSON format. If required, you can customize the report according to your needs using a JSON reader application.

Figure 7-65 Output of report_unique



The structure of the report is:

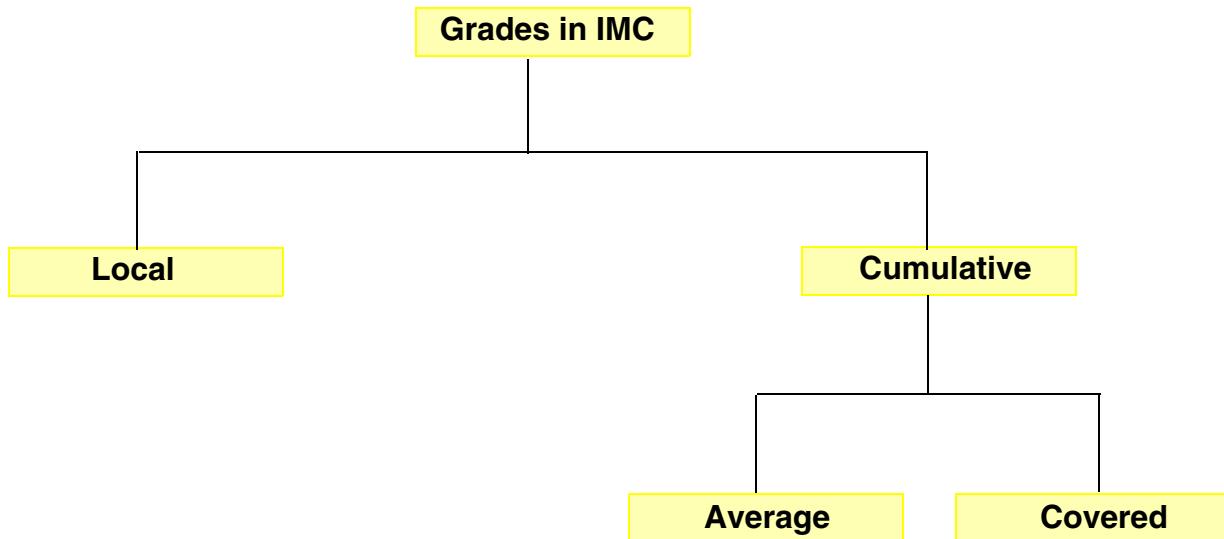
- The first container is the `command_input` container that shows the command options and parameters used to generate the report.
- After this, two group containers namely, `group_one_report` and `group_two_report` are shown. Within each of the group containers, there is a `unique_entities` container, which shows the unique items and their contributing runs.

For example, in the above report, unique item `sys/ascending/x/4` has total of 5 `hitting_runs`. However, out of these 5 runs, only three are listed because the `-uniq_run_count` was specified as 3. Notice that `count` is also listed for each run. It indicates the number of times that particular run was hit. The runs are listed in the ascending order due to the use of `-sort low_count` option.

7.3 Coverage Grading in IMC

[Figure 7-66 on page 407](#) shows the different grades calculated by IMC.

Figure 7-66 Grades in IMC



For each metrics type, IMC calculates the following grades:

- Local Grade: Local grade (also known as self coverage) is the coverage of that instance or module without considering the coverage of child instances or modules of that instance or module.
- Cumulative Grade: Cumulative grade is computed by combining the self coverage and the cumulative grade of its children, if any. Cumulative grade is of following types:
 - Cumulative Average Grade
 - Cumulative Covered Grade

For example, for block metrics, following grades are calculated by IMC:

- Block Local Grade

- Block Average Grade
- Block Covered Grade

In IMC GUI, these grades are shown as columns/attributes with the same name as stated above.

[Figure 7-67 on page 408](#) shows the grades columns related to block metrics in IMC GUI.

Figure 7-67 Block Local and Cumulative Grades in GUI Mode

Block Local Grade

Block Cumulative Average Grade Block Cumulative Covered Grade

Ex. Inv	Name	Block Average Grade	Block Covered Grade	Block Local Grade
	(no filter)	(no filter)	(no filter)	(no filter)
	Verification Metrics	77.6%	49.31%	n/a
	Types	75.34%	49.31%	n/a
	Instances	79.87%	49.31%	n/a
	dtmf_recv_core	79.87%	49.31%	n/a
	TEST_CONTROL_INST	n/a	n/a	n/a
	ROM_512x16_INST	n/a	n/a	n/a
	RAM_128x16_TEST_IN...	100%	100%	n/a
	RAM_256x16_TEST_IN...	100%	100%	n/a
	TDSP_DS_CS_INST	66.67%	66.67%	66.67%
	DATA_SAMPLE_MUX_I	100%	100%	100%

In batch mode, these grades are shown when you generate a summary report. For example, consider the following command:

```
report -summary -metrics block -cumulative on
```

[Figure 7-68 on page 409](#) shows the output of the above command.

Figure 7-68 Block Local and Cumulative Grades in Batch Mode

The diagram shows the output of the command `imc> report -summary -metrics block -cumulative on`. The output is a table with columns: name, Block*, Average, Block* Covered, and Block. The table lists various blocks and their metrics. Arrows point from the column headers to the corresponding data in the table. A vertical arrow points down to the 'Block Local Grade' label, which is positioned above the table. Two horizontal arrows point up to the 'Block Cumulative Average Grade' and 'Block Cumulative Covered Grade' labels, which are positioned below the table.

name	Block*	Average	Block* Covered	Block
dtmf_recv_r_core	80%	49% (431/873/14)	n/a	
- ROM_512x16_INST	n/a	n/a	n/a	
- RAM_128x16_TEST_INST	100%	100% (3/3/3)	n/a	
- RAM_128x16_INST	100%	100% (3/3/3)	100%	
- RAM_256x16_TEST_INST	100%	100% (3/3/3)	n/a	
- RAM_256x16_INST	100%	100% (3/3/3)	100%	
- TDSP_DS_CS_INST	67%	67% (4/6)	67%	
- DATA_SAMPLE_MUX_INST	100%	100% (9/9)	100%	
- DIGIT_REG_INST	67%	67% (2/3)	67%	
- RESULTS_CONV_INST	10%	10% (10/100)	10%	
- TDSP_MUX	100%	100% (3/3)	100%	

In the above report, * in the column name indicates cumulative grade. Cumulative grades are printed due to the use of `-cumulative` option in the `report` command. In the absence of this option, only local grade is printed.

For more details on reports, see [Generating Reports in GUI](#) on page 315.

This section discusses the following grade calculations for each metrics type in IMC:

- [Local Grade Calculation](#) on page 409
- [Cumulative Grade Calculation](#) on page 416

7.3.1 Local Grade Calculation

This section discusses local grade calculation for different metrics types.

7.3.1.1 Block Coverage

IMC uses the following formula to calculate block local grade:

$$\text{Block local grade} = \frac{\text{Number of covered blocks}}{\text{Total number of blocks}} \times 100\%$$

Note: In IMC GUI, the block local grade is printed in the *Block Local Grade* column. In batch mode, block local grade is printed when you use the `report -summary -metrics block` command.

Branch Coverage

Branch coverage is a subtype of block coverage and provides more precise coverage results by reporting coverage for each branch individually. IMC uses the following formula to calculate branch local grade:

$$\text{Branch local grade} = \frac{\text{Number of covered branches}}{\text{Total number of branches}} \times 100\%$$

Note: By default, branch coverage is not enabled. It is enabled using the `set_branch_scoring` command. See *ICC User Guide* for more details on this command. If branch coverage is enabled while scoring coverage, it is automatically reported when you generate a block summary report.

Statement Coverage

Statement coverage is a subtype of block coverage which provides information on the number of statements within a block. IMC uses the following formula to calculate statement local grade:

$$\text{Statement local grade} = \frac{\text{Number of statements in covered blocks}}{\text{Total number of statements in all blocks}} \times 100\%$$

Note: By default, statement coverage is not enabled. It is enabled using the `set_statement_scoring` command. See *ICC User Guide* for more details on this command. If statement coverage is enabled while scoring coverage, it is automatically reported when you generate a block summary report.

7.3.1.2 Expression Coverage

Expression coverage is calculated based on total number of min term rows in all of the expressions in an instance/module and the number of min term rows that are covered.

IMC uses the following formula to calculate the expression local grade:

$$\text{Expression local grade} = \frac{\text{Number of covered min term rows}}{\text{Total number of min term rows}} \times 100\%$$

In IMC, expression coverage is displayed at the following levels:

- Instance/Module
- Top expression
- Sub-expression
- Expression coverage table

For any given level, the total number of min term rows below that level is considered. For example, at the top expression level, min term rows of all the sub-expressions in the hierarchy of that expression are considered while calculating expression coverage of that top expression.

Note: In IMC GUI, the expression local grade is printed in the *Expression Local Grade* column. In batch mode, expression local grade is printed when you use the `report -summary -metrics expression` command.

7.3.1.3 Toggle Coverage

Toggle coverage is calculated based on total number of signal bits in an instance/module and the number of fully toggled signal bits. A signal bit is considered fully toggled signal if it has gone from 1 to 0 and also from 0 to 1.

IMC uses the following formula to calculate toggle local grade:

$$\text{Toggle local grade} = \frac{\text{Number of fully toggled signal bits}}{\text{Total number of signal bits}} \times 100\%$$

Note: In IMC GUI, the toggle local grade is printed in the *Toggle Local Grade* column. In batch mode, toggle local grade is printed when you use the `report -summary -metrics toggle` command.

7.3.1.4 State Coverage

IMC uses the following formula to calculate state local grade:

$$\text{State local grade} = \frac{\text{Number of visited states}}{\text{Total number of states}} \times 100\%$$

State coverage is calculated at two levels, instance/module level, and individual FSM level. While calculating state coverage at instance/module level, states of all FSMs in that instance/module are taken into consideration.

Note: In IMC GUI, the state local grade is printed in the *State Local Grade* column. In batch mode, state local grade is printed when you use the `report -summary -metrics state` command.

7.3.1.5 Transition Coverage

A transition means change of state from a start state to an end state. IMC uses the following formula to calculate transition local coverage:

$$\text{Transition local grade} = \frac{\text{Number of executed transitions}}{\text{Total number of transitions}} \times 100\%$$

Transition coverage is calculated at two levels, instance/module level, and individual FSM level. While calculating transition coverage at instance/module level, transitions of all FSMs in that instance/module are taken into consideration.

Note: In IMC GUI, the transition local grade is printed in the *Transition Local Grade* column. In batch mode, transition local grade is printed when you use the `report -summary -metrics transition` command.

7.3.1.6 Arc Coverage

Arc coverage is a subtype of transition coverage that reports all the input conditions under which a transition can take place. IMC uses the following formula to calculate arc local grade:

$$\text{Arc local grade} = \frac{\text{Number of executed arcs}}{\text{Total number of arcs}} \times 100\%$$

Arc coverage is calculated at two levels, instance/module level, and individual FSM level. While calculating arc coverage at instance/module level, the arcs of all FSMs in that instance/module are taken into consideration.

Note: If arc coverage is disabled for a particular FSM, then transition coverage is used as arc coverage for that FSM. By default, arc coverage is not enabled. It is enabled using the `set_fsm_arc_scoring` command. See *ICC User Guide* for more details on this command.

Note: In IMC GUI, the arc local grade is printed in the *Arc Local Grade* column. In batch mode, arc local grade is printed when you use the `report -summary -metrics arc` command.

7.3.1.7 Assertion Coverage

IMC uses the following formula to calculate assertion local grade:

$$\text{Assertion local grade} = \frac{\text{Number of finished assertions}}{\text{Total number of assertions}} \times 100\%$$

Note: In IMC GUI, the assertion local grade is printed in the *Assertion Local Grade* column. In batch mode, assertion local grade is printed when you use the `report -summary -metrics assertion` command.

7.3.1.8 Covergroup Coverage

A covergroup consists of one or more cover items (coverpoints or crosses) and a cover item includes one or more cover bins.

IMC uses the following formula to calculate coverage of a cover item:

$$\text{Cover item Covered coverage (C}_i\text{)} = \frac{\text{Number of covered cover bins}}{\text{Total number of cover bins}} \times 100\%$$

$$\text{If } \left\{ \frac{\text{Number of covered cover bins}}{\text{Total number of cover bins}} \times 100 = 100\% \right\}$$

$$\text{Cover item Average coverage (C}_i\text{)} = 100\%$$

else:

$$\text{Cover item Average coverage (C}_i\text{)} = \frac{\text{Number of covered cover bins}}{\text{Total number of cover bins} \times \text{goal}/100}$$

In the SystemVerilog language, a cover bin is considered covered if its hit count \geq the `at_least` count.

In the **e** language, a cover bin is considered:

- Fully covered—if its hit count \geq `at_least` count
- Partially covered—if its hit count is a non-zero value less than the `at_least` count

If a cover bin is partially covered, then the cover item coverage is calculated by replacing Number of covered bins in the above formula by hit count/at_least count.

IMC uses the following formula to calculate average coverage of a covergroup:

$$\text{Covergroup average coverage } (C_g) = \frac{\sum C_i \times W_i}{\sum W_i \times CG_g / 100}$$

In the above formula, C_i and W_i are the coverage and weight, respectively, of cover items in the covergroup C_g and CG_g is the coverage goal value of the Covergroup.

IMC uses the following formula to calculate covergroup average coverage of an instance/module:

$$\text{Covergroup average coverage of instance/module} = \frac{\sum C_g \times W_g}{\sum W_g \times IM_g / 100}$$

In the above formula, C_g and W_g are the coverage and weight, respectively, of covergroups in the instance/module and IM_g is the goal value of the instance/module. For details on the at_least count and weight of cover items, see the *ICC User Guide*.

IMC uses the following formula to calculate the *Covergroup Covered* grade:

$$\text{Covergroup Covered coverage} = \frac{\text{Number of covered cover bins}}{\text{Total number of cover bins}} \times 100\%$$

In the above formula, Total number of cover bins is the number of cover bins scored that have not been excluded.

Note: IMC does not consider weight/goal while calculating Number of covered cover bins and Total number of cover bins. The weight value affects only the average grade except when the weight is 0, which means exclude.

Note: In IMC GUI, the covergroup local grade is printed in the *CoverGroup Local Grade* column. In batch mode, covergroup local grade is printed when you use the report -summary -metrics covergroup command.

7.3.1.9 Code Coverage

Code coverage is computed by combining block coverage, expression coverage, and toggle coverage. IMC uses the following formula to calculate code local grade:

$$\text{Code local grade} = \frac{\text{Block coverage} + \text{Expression coverage} + \text{Toggle coverage}}{3}$$

Note: In IMC GUI, the code local grade is printed in the *Code Local Grade* column. In batch mode, code local grade is printed when you use the `report -summary -metrics code` command.

7.3.1.10 FSM Coverage

FSM coverage is computed by combining state coverage and transition coverage. IMC uses the following formula to calculate FSM local grade:

$$\text{FSM local grade} = \frac{\text{State coverage} + \text{Transition coverage}}{2}$$

Note: If arc coverage is enabled, then it is used in the above computation instead of transition coverage.

Note: In IMC GUI, the FSM local grade is printed in the *FSM Local Grade* column. In batch mode, FSM local grade is printed when you use the `report -summary -metrics fsm` command.

7.3.1.11 Functional Coverage

Functional coverage is computed by combining covergroup coverage and assertion coverage. IMC uses the following formula to calculate functional coverage:

$$\text{Functional local grade} = \frac{\text{Covergroup coverage} + \text{Assertion coverage}}{2}$$

Note: In IMC GUI, the functional local grade is printed in the *Functional Local Grade* column. In batch mode, functional local grade is printed when you use the `report -summary -metrics functional` command.

7.3.1.12 Overall Coverage

Overall coverage is computed by combining code coverage, FSM coverage, and functional coverage. IMC uses the following formula to calculate overall coverage:

$$\text{Overall local grade} = \frac{\text{Code coverage} + \text{FSM coverage} + \text{Functional coverage}}{3}$$

Note: In IMC GUI, the overall local grade is printed in the *Overall Local Grade* column. In batch mode, overall local grade is printed when you use the `report -summary -metrics overall` command.

7.3.2 Cumulative Grade Calculation

Cumulative grade is computed by combining the self metrics of that instance/type and the cumulative metrics of its children, if any.

The process by which two or more grades are combined is known as roll-up. IMC provides following grade roll-up methods:

- Average Rollup
- Covered Rollup

7.3.2.1 Average Rollup

The *Average* rollup method is used to calculate the Cumulative Average Grade of a metrics type. IMC uses the following formula to calculate cumulative average grade:

$$\text{Cumulative Average Grade} = \frac{M_{\text{self}} + \sum M_{\text{cum}}}{1 + \text{Number of immediate child instances}}$$

where M_{self} is the self metric (taking into consideration goal value, if defined) and M_{cum} is the cumulative metric of each of the immediate children while taking into consideration each weight and goal values.

Note: *Average* rollup method was earlier known as the *Top Elements* scheme.

7.3.2.2 Covered Rollup

The *Covered* rollup method is used to calculate the Cumulative Covered Grade of a metrics type. IMC uses the following formula to calculate cumulative covered grade:

$$\text{Cumulative Covered Grade} = \frac{\sum_{\text{Me_covered}} + \text{Descendents covered (i)}}{\sum_{\text{Me_total}} + \text{Descendents total (i)}}$$

where:

- Covered coverage of item i ($\text{covered}(i)$) is calculated as:

$$\text{covered}(i) = \sum_{\text{Children}} \text{Covered_bin}(i)$$

Here, $\text{covered_bin}(i)$ is the number of covered bins of item i .

- Total coverage of item i ($\text{total}(i)$) is calculated as:

$$\text{total}(i) = \sum_{\text{Children}} \text{bin}(i)$$

Here, $\text{bin}(i)$ is the number of bins of element i .

IMC does not consider `weight` while calculating covered grades.

Note: *Covered* rollup method was earlier known as the *All Buckets* scheme.

Incisive Metrics Center User Guide

List of Attributes

This chapter describes:

- [Attributes Associated with Blocks](#)
- [Attributes Associated with Expressions](#)
- [Attributes Associated with Toggle Variables](#)
- [Attributes Associated with Toggle Signals](#)
- [Attributes Associated with States](#)
- [Attributes Associated with Transitions](#)
- [Attributes Associated with Assertions](#)
- [Attributes Associated with Cover Bins](#)
- [Attributes Associated with Instances and Types](#)

A.1 Attributes Associated with Blocks

Attribute	Displays
Block Active	true if the block is active during simulation run. false if the block is inactive during simulation run.
Block Covered	1 if covered, 0 if uncovered
Block Average Grade	100% if covered, 0% is uncovered
Block Ignored	Number of blocks ignored
Block Total	Total number of blocks
Block Total Weighted Coverage	Weighted coverage of the block

Incisive Metrics Center User Guide

Attribute	Displays
Block Total Weights	Weight associated with the block
Block Type	Type of block
Block Uncovered	1 if uncovered, 0 if covered
Code Covered	1 if covered, 0 if uncovered
Code Average Grade	100% if covered, 0% is uncovered
Code Ignored	Number of blocks ignored
Code Total	Total number of blocks
Code Total Weighted Coverage	Weighted coverage of the block
Code Total Weights	Weight associated with the block
Code Uncovered	1 if uncovered, 0 if covered
Exclusion Comment	Blank, if block is not excluded or if no comment is added at the time of exclusion. Else, shows the value specified by the user.
Exclusion Reviewer	Blank, if block is not excluded. Unknown, if nothing is specified at the time of exclusion or if block is excluded without enabling adding comments at the time of exclusion. Else, shows the value specified by the user.
Exclusion Time	Blank, if block is not excluded. Else, shows the date and time when the exclusion is applied.
Exclusion User	Blank, if block is not excluded. Else, shows the login ID of the system on which the tool is running.

Incisive Metrics Center User Guide

Attribute	Displays
Exclusion Rule Type	<p>None if the block is not excluded</p> <p>Analysis Ungradable if the block is marked analysis ungradable</p> <p>Analysis Time if the block is excluded during analysis</p> <p>Simulation Time if the block is marked excluded during simulation run</p> <p>Non excludable if the block cannot be marked excluded</p> <p>Ungradable if the block is marked ungradable during simulation</p>
Source File	Path of the source file
Index	Unique identifier assigned to the block
Is Active	<code>true</code> if the block is active during simulation run <code>false</code> if the block is inactive during simulation run
Is Branch	<code>true</code> if the block is also a branch <code>false</code> if the block is not a branch
Source Start Line	Line number on which the block is defined
Name	Not Applicable
Origin File	Name of the file where block is defined
Origin Line	Line number where the block starts
Overall Covered	1 if covered, 0 if uncovered
Overall Average Grade	100% if covered, 0% if uncovered
Overall Ignored	Number of blocks ignored
Overall Total	Total number of blocks
Overall Total Weighted Coverage	Weighted overall coverage
Overall Total Weights	Weight associated with the block
Overall Uncovered	1 if uncovered, 0 if covered
Score	Hit count of the block
Statements	Number of statements in the selected block.

Attribute	Displays
Status	Covered if the block is covered during simulation run Not Covered if the block is uncovered during simulation run
UNR	F_UNR if the block is fully unreachable P_UNR if the block is partially unreachable E_UNR if the block is unreachable and marked excluded None if the block is not marked unreachable
Overall UNR	1 if block is marked unreachable, 0 if it is not marked unreachable.

A.2 Attributes Associated with Expressions

Attribute	Displays
Code Covered	Number of min term rows covered for the selected expression
Code Average Grade	Coverage grade for the selected expression
Code Ignored	Number of expressions ignored
Code Total	Number of min term rows
Code Total Weighted Coverage	Weighted coverage of the expression
Code Total Weights	Weight associated with the expression
Code Uncovered	Number of min term rows uncovered for the selected expression
Exclusion Comment	Blank, if expression is not excluded or if no comment is added at the time of exclusion. Else, shows the value specified by the user.

Incisive Metrics Center User Guide

Attribute	Displays
Exclusion Reviewer	Blank, if expression is not excluded. Unknown, if nothing is specified at the time of exclusion or if expression is excluded without enabling adding comments at the time of exclusion. Else, shows the value specified by the user.
Exclusion Time	Blank, if expression is not excluded. Else, shows the date and time when the exclusion is applied.
Exclusion User	Blank, if expression is not excluded. Else, shows the login ID of the system on which the tool is running.
Exclusion Rule Type	None if the expression is not excluded Analysis Time if the expression is excluded during analysis Simulation Time if the expression is marked excluded during simulation run Analysis Ungradable if the expression is marked analysis ungradable Non excludable if the expression cannot be marked excluded Ungradable if the expression is marked ungradable during simulation
Exp	The expression being evaluated
Expression Covered	Number of min term rows covered for the selected expression
Expression Average Grade	Coverage grade for the selected expression
Expression Ignored	Number of expressions ignored
Expression Total	Number of min term rows
Expression Total Weighted Coverage	Weighted coverage of the expression
Expression Total Weights	Weight associated with the expression

Incisive Metrics Center User Guide

Attribute	Displays
Expression Uncovered	Number of min term rows uncovered for the selected expression
Source File	Path of the source file
Index	Unique identification number assigned to the expression
Is Active	<code>true</code> if the expression is active during simulation run <code>false</code> if the expression is inactive during simulation run
Source Start Line	Line number where the expression is defined
Name	Not applicable
Overall Covered	Number of min term rows covered for the selected expression
Overall Average Grade	Coverage grade for the selected expression
Overall Ignored	Number of expressions ignored
Overall Total	Number of min term rows
Overall Total Weighted Coverage	Weighted coverage of the expression
Overall Total Weights	Weight associated with the expression
Overall Uncovered	Number of min term rows uncovered for the selected expression
UNR	<code>F_UNR</code> if the expression is fully unreachable <code>P_UNR</code> if the expression is partially unreachable <code>E_UNR</code> if the expression includes some min-terms that are marked as unreachable, but all of these min-terms are excluded <code>None</code> if the expression is not marked unreachable
Overall UNR	Shows the number of min-term rows of the expression that are marked as unreachable. Shows 0 if the min term is not marked unreachable.

A.3 Attributes Associated with Toggle Variables

Attribute	Displays
Code Covered	Number of signals covered for selected variable
Code Average Grade	Coverage grade for the selected toggle variable
Code Ignored	Number of toggle variables ignored
Code Total	Number of signals for selected variable
Code Total Weighted Coverage	Weighted coverage of the toggle variable
Code Total Weights	Weight associated with the toggle variable
Code Uncovered	Number of uncovered signals of selected variable
Exclusion Comment	Blank, if variable is not excluded or if no comment is added at the time of exclusion. Else, shows the value specified by the user.
Exclusion Reviewer	Blank, if variable is not excluded. Unknown, if nothing is specified at the time of exclusion or if the variable is excluded without enabling adding comments at the time of exclusion. Else, shows the value specified by the user.
Exclusion Time	Blank, if variable is not excluded. Else, shows the date and time when the exclusion is applied.
Exclusion User	Blank, if variable is not excluded. Else, shows the login ID of the system on which the tool is running.

Incisive Metrics Center User Guide

Attribute	Displays
Exclusion Rule Type	<p>None if the variable is not excluded</p> <p>Analysis Time if the variable is excluded during analysis</p> <p>Simulation Time if the variable is marked excluded during simulation run</p> <p>Analysis Ungradable if the variable is marked analysis ungradable</p> <p>Non excludable if the variable cannot be marked excluded</p> <p>Ungradable if the variable is marked ungradable during simulation</p>
Source File	Path of the source file
Is Active	<p>true if the variable is active during simulation run</p> <p>false if the variable is inactive during simulation run</p>
Source Start Line	Line number where the variable is defined
Name	Name of the variable
Kind	<p>Kind of variable. It can have any of the following values:</p> <ul style="list-style-type: none"> ■ input—for input port signals ■ inout—for inout port signals ■ output—for output port signals ■ internal—for internal wires / regs <p>Using this attribute, you can filter the toggle variables based on the variable type (input/inout/output/internal). In addition, you can also generate reports based on a particular type of variables.</p>
Overall Covered	Number of signals covered for selected variable
Overall Average Grade	Coverage grade for the selected toggle variable
Overall Ignored	Number of toggle variables ignored
Overall Total	Number of signals for selected variable
Overall Total Weighted Coverage	Weighted coverage of the toggle variable

Attribute	Displays
Overall Total Weights	Weight associated with the toggle variable
Overall Uncovered	Number of uncovered signals of selected variable
Range	The range of the variable.
Toggle Covered	Number of signals covered for selected variable
Toggle Average Grade	Coverage grade for the selected toggle variable
Toggle Ignored	Number of toggle variables ignored
Toggle Total	Number of signals for selected variable
Toggle Total Weighted Coverage	Weighted coverage of the toggle variable
Toggle Total Weights	Weight associated with the toggle variable
Toggle Uncovered	Number of uncovered signals of selected variable
UNR	F_UNR if the toggle variable is fully unreachable P_UNR if the toggle variable is partially unreachable E_UNR – If the toggle variable includes some toggle signals that are marked as unreachable, but all of these signals are excluded None if the toggle variable is not marked unreachable
Overall UNR	Number of signals within the toggle variable that are marked unreachable.

A.4 Attributes Associated with Toggle Signals

Attribute	Displays
Code Covered	1 if covered, 0 if uncovered
Code Average Grade	100% if covered, 0% is uncovered
Code Ignored	Number of toggle signals ignored
Code Total	Number of signals for selected variable
Code Total Weighted Coverage	Weighted coverage of the toggle signal

Incisive Metrics Center User Guide

Attribute	Displays
Code Total Weights	Weight associated with the toggle signal
Code Uncovered	1 if uncovered, 0 if covered
Count Tx to One	Number of times the signal rise (0->1) during simulation run
Count Tx to Zero	Number of times signal fall (1->0) during simulation run
Exclusion Comment	Blank, if signal is not excluded or if no comment is added at the time of exclusion. Else, shows the value specified by the user.
Exclusion Reviewer	Blank, if signal is not excluded. Unknown, if nothing is specified at the time of exclusion or if signal is excluded without enabling adding comments at the time of exclusion. Else, shows the value specified by the user.
Exclusion Time	Blank, if signal is not excluded. Else, shows the date and time when the exclusion is applied.
Exclusion User	Blank, if signal is not excluded. Else, shows the login ID of the system on which the tool is running.
Exclusion Rule Type	None if the signal is not excluded Analysis Time if the signal is excluded during analysis Simulation Time if the signal is marked excluded during simulation run Analysis Ungradable if the signal is marked analysis ungradable Non excludable if the signal cannot be marked excluded Ungradable if the signal is marked ungradable during simulation
Is Active	true if the signal is active during simulation run false if the signal is inactive during simulation run
Name	Name of the signal
Overall Covered	1 if covered, 0 if uncovered

Incisive Metrics Center User Guide

Attribute	Displays
Overall Average Grade	100% if covered, 0% is uncovered
Overall Ignored	Number of toggle signals ignored
Overall Total	Number of signals for selected variable
Overall Total Weighted Coverage	Weighted coverage of the toggle signal
Overall Total Weights	Weight associated with the toggle signal
Overall Uncovered	1 if uncovered, 0 if covered
Score	Hit count of the signal
Status	Covered if the signal bits toggled through rise as well as fall transitions at least once. Not Covered if the signal bits did not toggle through both rise as well as fall transitions at least once.
Toggle Covered	1 if covered, 0 if uncovered
Toggle Average Grade	100% if covered, 0% is uncovered
Toggle Ignored	Number of toggle signals ignored
Toggle Total	Number of signals for selected variable
Toggle Total Weighted Coverage	Weighted coverage of the toggle signal
Toggle Total Weights	Weight associated with the toggle signal
Toggle Uncovered	1 if uncovered, 0 if covered
UNR	F_UNR if one or both (rise and fall) transitions are marked as unreachable E_UNR – If the toggle signal is excluded and marked as unreachable None if none of its transition is marked unreachable
Overall UNR	0 if none of its bits (fall or rise) are marked as unreachable 1 if at least one of its bits (fall or rise) are marked as unreachable

A.5 Attributes Associated with States

Attribute	Displays
Encoding	Encoding value of the state
Exclusion Comment	Blank, if state is not excluded or if no comment is added at the time of exclusion. Else, shows the value specified by the user.
Exclusion Reviewer	Blank, if state is not excluded. Unknown, if nothing is specified at the time of exclusion or if state is excluded without enabling adding comments at the time of exclusion. Else, shows the value specified by the user.
Exclusion Time	Blank, if state is not excluded. Else, shows the date and time when the exclusion is applied.
Exclusion User	Blank, if state is not excluded. Else, shows the login ID of the system on which the tool is running.
Exclusion Rule Type	None if the state is not excluded Analysis Smart if the state is smart excluded during analysis Analysis Smart Indirect if the state got excluded indirectly due to a <i>Exclude Smart</i> action Analysis Time if the state is excluded during analysis Simulation Time if the state is marked excluded during simulation run Analysis Ungradable if the state is marked analysis ungradable Non excludable if the state cannot be marked excluded Ungradable if the state is marked ungradable during simulation
FSM Covered	1 if covered, 0 if uncovered

Incisive Metrics Center User Guide

Attribute	Displays
FSM Average Grade	100% if covered, 0% is uncovered
FSM Ignored	Number of states ignored
FSM Total	Total number of states
FSM Total Weighted Coverage	Weighted coverage of the state
FSM Total Weights	Weight associated with the state
FSM Uncovered	1 if uncovered, 0 if covered
Source File	Path of the source file
Is Active	<code>true</code> if the state is active during simulation run <code>false</code> if the state is inactive during simulation run
Is Reset State	<code>true</code> if the state is a reset state <code>false</code> if the state is not a reset state
Source Start Line	Line number where the state is defined
Name	Name of the state
Overall Covered	1 if covered, 0 if uncovered
Overall Average Grade	100% if covered, 0% is uncovered
Overall Ignored	Number of states ignored
Overall Total	Total number of states
Overall Total Weighted Coverage	Weighted coverage of the state
Overall Total Weights	Weight associated with the state
Overall Uncovered	1 if uncovered, 0 if covered
Reset Count	Reset count of the state
Score	Visit count of the state
State Covered	1 if covered, 0 if uncovered
State Average Grade	100% if covered, 0% is uncovered
State Ignored	Number of states ignored
State Total	Total number of states

Attribute	Displays
State Total Weighted Coverage	Weighted coverage of the state
State Total Weights	Weight associated with the state
State Uncovered	1 if uncovered, 0 if covered
Status	Covered if the state is covered during simulation run Not Covered if the state is uncovered during simulation run

A.6 Attributes Associated with Transitions

Attribute	Displays
Exclusion Comment	Blank, if transition is not excluded or if no comment is added at the time of exclusion. Else, shows the value specified by the user.
Exclusion Reviewer	Blank, if transition is not excluded. Unknown, if nothing is specified at the time of exclusion or if transition is excluded without enabling adding comments at the time of exclusion. Else, shows the value specified by the user.
Exclusion Time	Blank, if transition is not excluded. Else, shows the date and time when the exclusion is applied.
Exclusion User	Blank, if transition is not excluded. Else, shows the login ID of the system on which the tool is running.

Incisive Metrics Center User Guide

Attribute	Displays
Exclusion Rule Type	<p>None if the transition is not excluded</p> <p>Analysis Smart if the transition is smart excluded during analysis</p> <p>Analysis Smart Indirect if the transition got excluded indirectly due to a <i>Exclude Smart</i> action</p> <p>Analysis Time if the transition is excluded during analysis</p> <p>Simulation Time if the transition is marked excluded during simulation run</p> <p>Analysis Ungradable if the transition is marked analysis ungradable</p> <p>Non excludable if the transition cannot be marked excluded</p> <p>Ungradable if the transition is marked ungradable during simulation</p>
FSM Covered	1 if covered, 0 if uncovered
FSM Average Grade	100% if covered, 0% is uncovered
FSM Ignored	Number of transitions ignored
FSM Total	Total number of transitions
FSM Total Weighted Coverage	Weighted coverage of the transition
FSM Total Weights	Weight associated with the transition
FSM Uncovered	1 if uncovered, 0 if covered
From State File	Path of the source file
From State Line	110
From State Name	Name of the from state
Index	Transition identifier number
Is Active	<p>true if the transition is active during simulation run</p> <p>false if the transition is inactive during simulation run</p>

Incisive Metrics Center User Guide

Attribute	Displays
Is Reset Trans	true if the transition is the reset transition false if the transition is not the reset transition
Name	Not applicable
Overall Covered	1 if covered, 0 if uncovered
Overall Average Grade	100% if covered, 0% if uncovered
Overall Ignored	Number of transitions ignored
Overall Total	Total number of transitions
Overall Total Weighted Coverage	Weighted coverage of the transition
Overall Total Weights	Weight associated with the transition
Overall Uncovered	1 if uncovered, 0 if covered
Score	Visit count of the transition
Status	Covered if the transition is visited even once Not Covered if the transition is not visited even once
To State File	Path of the source file
To State Line	Line number of the To State
To State Name	Name of the To State
Transition Covered	1 if covered, 0 if uncovered
Transition Average Grade	100% if covered, 0% if uncovered
Transition Ignored	Number of transitions ignored
Transition Total Weighted Coverage	Total number of transitions
Transition Total Weights	Weighted coverage of the transition
Transition Total	Weight associated with the transition
Transition Uncovered	1 if uncovered, 0 if covered

A.7 Attributes Associated with Assertions

Attribute	Displays
Assertion Covered	0 is the assertion is not covered during simulation run 1 if the assertion is covered during simulation run
Assertion Average Grade	100% if covered, 0% is uncovered
Assertion Ignored	Number of assertions ignored
Assertion Total	Total number of assertions
Assertion Total Weighted Coverage	Weighted coverage of the assertion
Assertion Total Weights	Weight associated with the assertion
Assertion Uncovered	1 if uncovered, 0 if covered
Enclosing Entity	Hierarchical name of the entity in which assertion is defined
Exclusion Comment	Blank, if assertion is not excluded or if no comment is added at the time of exclusion. Else, shows the value specified by the user.
Exclusion Reviewer	Blank, if assertion is not excluded. Unknown, if nothing is specified at the time of exclusion or if assertion is excluded without enabling adding comments at the time of exclusion. Else, shows the value specified by the user.
Exclusion Time	Blank, if assertion is not excluded. Else, shows the date and time when the exclusion is applied.
Exclusion User	Blank, if assertion is not excluded. Else, shows the login ID of the system on which the tool is running.

Incisive Metrics Center User Guide

Attribute	Displays
Exclusion Rule Type	<p>None if the assertion is not excluded</p> <p>Analysis Time if the assertion is excluded during analysis</p> <p>Simulation Time if the assertion is marked excluded during simulation run</p> <p>Analysis Ungradable if the assertion is marked analysis ungradable</p> <p>Non excludable if the assertion cannot be marked excluded</p> <p>Ungradable if the assertion is marked ungradable during simulation</p>
Failed Count	Number of times the assertion failed
Source File	Path of source file
Finished Count	Number of times the assertion finished successfully
Vacuous Count	<p>If the -abvrecordvacuous switch is used at the time of simulation run, then it shows the number of times the assertion was vacuous pass but never finished.</p> <p>If the -abvrecordvacuous switch is not used at simulation, then it shows n/a.</p>
Disabled Count	<p>If the -abvrecordvacuous switch is used at the time of simulation run, then it shows the number of times the assertion transitioned to the disabled state but never finished.</p> <p>If the -abvrecordvacuous switch is not used at simulation, then it shows n/a.</p>
Attempt Count	<p>If the -abvrecordvacuous switch is used at the time of simulation run, then it shows the number of times the assertion was attempted.</p> <p>If the -abvrecordvacuous switch is not used at simulation, then it shows n/a.</p>
Functional Covered	<p>0 if the assertion is not covered during simulation run</p> <p>1 if the assertion is covered during simulation run</p>
Functional Average Grade	100% if covered, 0% is uncovered

Incisive Metrics Center User Guide

Attribute	Displays
Functional Ignored	Number of assertions ignored
Functional Total	Total number of assertions
Functional Total Weighted Coverage	Weighted coverage of the assertion
Functional Total Weights	Weight associated with the assertion
Functional Uncovered	1 if uncovered, 0 if covered
Is Active	<code>true</code> if the assertion is active during simulation run <code>false</code> if the assertion is inactive during simulation run
Source Line	Line number where the assertion is defined
Status	<p>Passed if the assertion finished without failure at least once, and never failed.</p> <p>Failed if the assertion finished with failure at least once.</p> <p>Other if the assertion never finished.</p> <p>Note: If the <code>-abvrecordvacuous</code> switch is used at the time of simulation run, then following values can also show:</p> <ul style="list-style-type: none"> ■ Vacuous if the assertion was vacuous pass at least once but never finished. ■ Disabled if the assertion transitioned to the disabled state at least once but never finished. ■ Attempted if the assertion was attempted at least once but never finished. ■ Not Attempted if the assertion was not attempted all.
Name	Name of the assertion
Overall Covered	0 if the assertion is not covered during simulation run 1 if the assertion is covered during simulation run
Overall Average Grade	100% if covered, 0% if uncovered
Overall Ignored	Number of assertions ignored
Overall Total	Total number of assertions

Incisive Metrics Center User Guide

Attribute	Displays
Overall Total Weighted Coverage	Weighted coverage of the assertion
Overall Total Weights	Weight associated with the assertion
Overall Uncovered	1 if uncovered, 0 if covered
Proof Bound	It is the value set by the user (or propagated from a parent port/section) for the assertion.
Actual Proof Bound	It is the actual bound used for proving the assertion. It is the bound of a formal property with bounded proof status in the formal environment.
Formal Total	Total number of formal assertions.
Formal Status Grade	100% if assertion is covered by the formal run, else 0%.
Formal Status	Proved if the assertion is proved. Failed if the assertion finished with failure at least once. Other if the assertion never finished.
Formal Proved Count	1 if the assertion is Proved, else 0.
Formal Proved	1.0 if the assertion is Proved, else 0.0.
Formal Passed Count	1 if the assertion is passed, else 0.
Formal Other	1.0 if the assertion status is Other, else 0.0.
Formal Finished Count	1 if the assertion is finished successfully, else 0.
Formal Failed Count	1 if the assertion is failed, else 0.
Formal Failed	1.0 if the assertion is failed, else 0.0.
Formal Covered	0 if the assertion is not covered by formal run 1 if the formal assertion is covered by formal run
Formal Cover Failed Count	1 if the assertion was marked unreachable by formal run, else 0.
Formal Average Grade	100% if covered, 0% is uncovered.

Attribute	Displays
UNR	F_UNR if the assertion is fully unreachable P_UNR if the assertion is partially unreachable E_UNR if the assertion is unreachable and marked excluded None if the assertion is not marked unreachable

A.8 Attributes Associated with Cover Bins

Attribute	Displays
Atleast	Value of the <code>at_least</code> option
Cover Item	Hierarchical name of the cover item in which the bin is defined
CoverGroup Covered	1 if covered, 0 if uncovered
CoverGroup Average Grade	100% if covered, 0% is uncovered
CoverGroup Ignored	1 if ignored, 0 if not ignored
CoverGroup Total	Total number of covergroups
CoverGroup Total Weighted Coverage	Weighted coverage of the cover bin
CoverGroup Total Weights	Weight associated with the cover bin
CoverGroup Uncovered	1 if uncovered, 0 if covered
Exclusion Comment	Blank, if bin is not excluded or if no comment is added at the time of exclusion. Else, shows the value specified by the user.
Exclusion Reviewer	Blank, if bin is not excluded. Unknown, if nothing is specified at the time of exclusion or if bin is excluded without enabling adding comments at the time of exclusion. Else, shows the value specified by the user.

Incisive Metrics Center User Guide

Attribute	Displays
Exclusion Time	Blank, if bin is not excluded. Else, shows the date and time when the exclusion is applied.
Exclusion User	Blank, if bin is not excluded. Else, shows the login ID of the system on which the tool is running.
Exclusion Rule Type	None if the bin is not excluded Analysis Smart if the bin is smart excluded during analysis Analysis Smart Indirect if the bin got excluded indirectly due to a <i>Exclude Smart</i> action Analysis Time if the bin is excluded during analysis Simulation Time if the bin is marked excluded during simulation run Analysis Ungradable if the bin is marked analysis ungradable Non excludable if the bin cannot be marked excluded Ungradable if the bin is marked ungradable during simulation
Source File	Path of the source file
Functional Covered	1 if covered, 0 if uncovered
Functional Average Grade	100% if covered, 0% is uncovered
Functional Ignored	1 if ignored, 0 if not ignored
Functional Total	Total number of covergroups
Functional Total Weighted Coverage	Weighted coverage of the cover bin
Functional Total Weights	Weight associated with the cover bin
Functional Uncovered	1 if uncovered, 0 if covered
Is Active	true if the bin is active during simulation run false if the bin is inactive during simulation run
Score	Hit count of the cover bin

Attribute	Displays
Source Start Line	Line number on which the bin is defined
Name	Name of the bin
Overall Covered	1 if covered, 0 if uncovered
Overall Average Grade	100% if covered, 0% is uncovered
Overall Ignored	1 if ignored, 0 if not ignored
Overall Total	Total number of covergroups
Overall Total Weighted Coverage	Weighted coverage of the cover bin
Overall Total Weights	Weight associated with the cover bin
Overall Uncovered	1 if uncovered, 0 if covered

A.9 Attributes Associated with Instances and Types

The attributes related to block, expressions, signals, assertions, and covergroups also show when you select an instance or a type on the summary page. These are already discussed in above sections. In addition to the attributes discussed in above sections, a few more attributes are shown when you select an instance or a type. This section will discuss only the

Incisive Metrics Center User Guide

attributes that are related to instances and types and the ones that are not discussed in above sections.

Attribute	Displays
Assertion Status Grade	<p>Is the total percentage of passed assertions out of all assertions, which is calculated as</p> $<\text{TotalPassedAssertions}> / <\text{AssertionTotal}> * 100$ <p>where</p> <ul style="list-style-type: none">■ <code>TotalPassedAssertions</code> is the total number of assertions that finished without failure at least once, and never failed.■ <code>AssertionTotal</code> is the total number of assertions, which is the sum of <code>TotalPassedAssertions</code>, <code>TotalFailedAssertions</code>, and <code>OtherAssertions</code>. <p>Note: <code>OtherAssertions</code> is the total number of assertions that never finished and is calculated as $[\text{AssertionTotal} - (\text{TotalPassedAssertions} + \text{TotalFailedAssertions})]$</p>
Full Path	<p>The complete path of the selected instance.</p> <p>This attribute applies only to instances.</p>
Is Instantiated	<p>This attribute applies only to types.</p> <p><code>true</code> if the selected item is a class and is instantiated.</p> <p><code>false</code> if the selected item is a class and is not instantiated.</p> <p>Not applicable if the selected item is a type and not a class.</p>
Language	Is the language of the underlying source code, for example, Verilog, VHDL, e.

Incisive Metrics Center User Guide

Attribute	Displays
Assertion Local Status Grade	<p>Is the total percentage of local passed assertions out of all assertions, which is calculated as</p> $<\text{TotalLocalPassedAssertions}> / <\text{AssertionLocalTotal}> * 100$ <p>where</p> <ul style="list-style-type: none"> ■ <code>TotalLocalPassedAssertions</code> is the total number of local assertions that finished without failure at least once, and never failed. ■ <code>AssertionLocalTotal</code> is the total number of local assertions, which is the sum of <code>TotalLocalPassedAssertions</code>, <code>TotalLocalFailedAssertions</code>, and <code>LocalOtherAssertions</code>. <p>Note: <code>LocalOtherAssertions</code> is the total number of local assertions that never finished and is calculated as $[\text{AssertionLocalTotal} - (\text{TotalLocalPassedAssertions} + \text{TotalLocalFailedAssertions})]$</p>
Overall Covered Grade	Overall covered grade of the type or instance.
Overall Local Covered	Total number of local covered items.
Overall Local Grade	Overall covered grade of the type or instance.
Overall Local Ignored	Total number of local ignored items.
Overall Local Total	Total number of local coverage items.
Overall Local Total Weighted Coverage	Weighted coverage of the type or instance.
Overall Local Total Weights	Weight associated with the type or instance.
Overall Local UNR	Number of local unreachable items/ All local entities Note: Here, all entities include code coverage only.
Overall UNR	Number of cumulative unreachable items/ All cumulative entities Note: Here, all entities include code coverage only.
Overall Local Uncovered	Total number of local uncovered items.

Attribute	Displays
Assertion Failed	Number of non excluded assertions in sub-hierarchy with Failed status.
Assertion Local Failed	Number of non excluded local assertions with Failed status.
Assertion Local Passed	Number of non excluded local assertions with Passed status.
Assertion Passed	Number of non excluded assertions in sub-hierarchy with Passed status.
Assertion Local Other	Number of non excluded local assertions with Other status.
Assertion Other	Number of non excluded assertions in sub-hierarchy with Other status.
Formal Uncovered	It is the total number of uncovered formal assertions.
Formal Total	It is the total number of formal assertions.
Formal Status Grade	<p>Is the total percentage of proved formal assertions out of all formal assertions. The <i>Formal Status Grade</i> is presented in the <i>Assertion</i> table (in relevant views) as a status bar with following colors:</p> <ul style="list-style-type: none"> ■ Green —for Proved assert properties ■ Red — for Failed assert properties ■ Gray — for Other assert properties <p>The value Proved can be controlled by using the <i>Check Must Trace</i> configuration option. For more details, see Configuring Formal Analysis Options on page 84.</p>
Formal Proved	It is the number of formal assertions with status Proved.
Formal Other	It is the number of formal assertions with status Other.
Formal Failed	It is the number of formal assertions with status Failed.
Formal Total	It is the total number of formal properties that affect the formal grade (unexcluded assert / cover properties). It does not include properties which do not affect the formal grade (assume properties).
Formal Covered	It is the number of formal covered assertions (It can be assertions with status Other or Proved).

Incisive Metrics Center User Guide

Attribute	Displays
Formal Average Grade	<p>It is the calculated as:</p> <p>Formal Covered / Formal Total</p>
Type File	<p>Is the complete path of the file in which the module (type) of the selected instance is defined.</p> <p>This attribute applies only to instances.</p>
Type Line	<p>Is the line number on which the module (type) of the selected instance is defined.</p> <p>This attribute applies only to instances.</p>
Type Name	<p>Is the name of the module (type) of the selected instance.</p> <p>This attribute applies only to instances.</p>
Exclusion Rule Type	<p>None if the instance or type is not excluded</p> <p>Analysis Ungradable if the instance or type is marked analysis ungradable</p> <p>Analysis Time if the instance or type is excluded during analysis</p> <p>Simulation Time if the instance or type is marked excluded during simulation run</p> <p>Non excludable if the instance or type cannot be marked excluded.</p> <p>Ungradable if the instance or type is marked ungradable during simulation</p> <p>Dirty children for the parent of the instance or type with invalid marking.</p> <p>Dirty Self for the instance or type with invalid marking</p>
UNR	<p>F_UNR if all of the instance/module entities are marked as unreachable (blocks/expressions/toggles/assertions).</p> <p>P_UNR if some of the instance/module entities are marked as unreachable.</p> <p>E_UNR if some of the instance/module entities are marked as unreachable, but all of these entities are excluded.</p> <p>None if none of the instance/module entities are marked as unreachable.</p>

Incisive Metrics Center User Guide

Known Gaps and Differences

In this release, IMC will not replace ICCR and the Specman coverage GUI. There are a few gaps that will be addressed over releases. Once the key gaps are addressed, Cadence plans to completely replace ICCR and the Specman coverage GUI.

This chapter discusses:

- [Known Gaps in IMC in Comparison with ICCR](#)
- [Known Gaps in IMC in Comparison with the Specman Coverage GUI](#)
- [Differences from ICCR Functionality](#)

Note: Currently, there are no automated batch command scripts to migrate to IMC.

B.1 Known Gaps in IMC in Comparison with ICCR

- IMC does not support GUI invocation from the batch mode.
- IMC does not support loading of multiple runs (in batch as well as GUI mode). ICCR supports loading multiple runs that share the same model.
- IMC does not support merging and reporting of Technology Independent Contribution (TIC) generated by Incisive.

B.2 Known Gaps in IMC in Comparison with the Specman Coverage GUI

- IMC does not support switching between simulation and analysis. In the Specman coverage GUI, you can load the e code, and analyze the coverage model simultaneously, along with code development. In IMC, the coverage model is available for analysis only after simulation run.
- IMC does not support loading multiple runs. You need to merge runs (in the batch mode) and then load the merged run to achieve overall coverage.

- IMC does not support ranking in GUI mode.
- IMC does not show details related to contribution of elements in the tests. In the Specman coverage GUI, you can view the contributing runs for each element.



At this time, features such as merging and ranking, are supported only in the batch mode of IMC.

B.3 Differences from ICCR Functionality

This section describes the key differences between IMC and ICCR.

Table B-1 Loading and Listing Runs

ICCR	IMC
Supports loading multiple tests.	Does not support loading multiple runs.
Supports listing all loaded tests using the <code>list_coverage_file</code> command.	Supports listing the currently loaded run using the <code>show</code> command.
Supports unloading all loaded tests using the <code>reset_coverage</code> command.	Supports unloading the loaded run with the <code>unload</code> command.
Supports listing modules/instances with enabled coverage types using the <code>list_coverage</code> command.	Supports listing types/instances with enabled coverage types using the <code>report -list</code> command.
Supports listing all modules or entities in the design using the <code>list_modules</code> or the <code>list_design_entities</code> command.	Supports listing all types along with enabled coverage types using the <code>report -list-type</code> command.
Supports listing all instances in the design using the <code>list_instances</code> command.	Supports listing all instances along with enabled coverage types using the <code>report -list</code> command.
Supports listing all FSMs in the design using the <code>list_fsm</code> command.	Supports listing types or instances in which FSMs are scored using the <code>report -list-metrics fsm</code> command.

Note: By default, ICCR is launched as an analysis tool from Enterprise Manager. To launch IMC from Enterprise Manager, see *Invoking IMC* in the *Incisive Enterprise Manager Analyzing Metrics* guide.

Table B-2 Reporting

ICCR	IMC
Supports summary report using the <code>report_summary</code> command.	Supports summary report using the <code>report_summary</code> command. However, currently, does not allow you to disable printing of covergroup options.
Supports detailed report using the <code>report_detail</code> command.	Supports detailed report using the <code>report_detail</code> command. However, currently, does not allow you to: <ul style="list-style-type: none">■ Disable printing of covergroup options.■ Enable listing of consecutive covergroup bins in separate rows. Note: You may see a difference in the overall coverage number and cumulative overall coverage number in the detailed report of IMC as compared to ICCR. The numbers generated by IMC match with the ICCR HTML report.
Prints indices in the detailed report only if explicitly turned on using the <code>indices</code> command.	By default, prints indices in the detailed report.
Does not generate indices for sub-expressions in an expression coverage detailed report.	Generates indices for all sub-expressions.
Generate blocks are not listed separately in a module-based report or in the <i>Module</i> list shown in the <i>Code/Data</i> tab page of the GUI. The coverage of generate blocks is rolled-up to the parent when calculating the self/cumulative coverage of the module.	Generate blocks are listed separately in a type-based report or below the <i>Types</i> node in the <i>Verification Hierarchy</i> pane of the GUI. Therefore, coverage of generate blocks is not rolled-up to the parent type when calculating coverage grade of the type that includes generate blocks.

ICCR	IMC
Supports tabular summary report using the <code>report_tabular_summary</code> command.	Does not support the tabular summary report.
Supports block-annotated source report using the <code>report_block_annotated_source</code> command.	Does not support the block-annotated source report.
Supports HTML report using the <code>report_html</code> command.	Supports HTML report using the <code>report -html</code> command.
Allows deselecting coverage types for the specified instance or module/design entity in currently loaded tests using the <code>deselect_coverage</code> command.	Currently, not supported.
Supports selecting coverage types for the specified instance or module/design entity (previously deselected using the <code>deselect_coverage</code> command) in the currently loaded tests using the <code>select_coverage</code> command.	Currently, not supported.
Supports launching of GUI from <code>iccr</code> prompt using the <code>view_graphics</code> command.	Currently, not supported.

Table B-3 Merging

ICCR	IMC
Supports merging as a three-step process: <ul style="list-style-type: none"> ■ Set the DUT for merge using the <code>set_dut_modules</code> command (optional). ■ Specify the merge behavior using the <code>set_merge</code> command (optional). ■ Merge the tests using the <code>merge</code> command. 	Supports merging as a two-step process: <ul style="list-style-type: none"> ■ Set the merge configuration, such as DUT, instances, and mode of merge using the <code>merge_config</code> command (optional). ■ Merge the runs using the <code>merge</code> command.

ICCR	IMC
Merge conflict items are shown in detailed report.	Currently, merge conflict items are not shown in IMC reports.

Table B-4 Ranking

ICCR	IMC
Ranks the tests using the <code>test_order</code> command.	Supports ranking as a two-step process: <ul style="list-style-type: none"> ■ Set the ranking configuration using the <code>rank_config</code> command (optional). ■ Rank the runs using the <code>rank</code> command.
Supports identifying recent test enhancements using the <code>difference</code> command.	Does not support identifying recent test enhancements.
Supports identifying duplicates across tests using the <code>intersection</code> command.	Does not support identifying duplicates across tests.
Supports generating cumulative numbers from tests using the <code>union</code> command.	Does not support generating cumulative numbers.

Table B-5 Marking

ICCR	IMC
Supports marking a particular block or expression using the <code>mark</code> command. Note: ICCR supports marking of only blocks and expressions.	Supports refinement of all metric types and coverage items using the <code>exclude</code> command.
Supports unmarking the marked items using the <code>undo_mark</code> command.	Supports un-excluding the excluded items using the <code>unexclude</code> command.
Supports saving applied marks for reuse later using the <code>save_icf</code> command.	Supports saving refinements using the <code>save-refinement</code> command.

ICCR	IMC
The marks applied using the <code>mark/undo_mark</code> command or using the ICCR GUI are saved in form of exact commands in the ICCR configuration file.	The refinements applied using the <code>exclude/unexclude</code> command or IMC GUI are not saved in form of exact commands in the <code>vRefine</code> file. Instead, they are saved as exclusion rules. These rules are different from batch-mode <code>exclude</code> and <code>unexclude</code> commands.
Supports loading of saved marks using the <code>load_icf</code> command.	Supports loading the refinements file using the <code>load -refinement</code> command.

Note: IMC provides a command (`convert_icf`) using which you can convert an ICCR Configuration File (ICF) into an IMC exclude file. For more details, see [Converting an ICF to an IMC Exclude File](#) on page 232.

Index

A

add columns [64](#)
advanced filtering [70](#)
advanced item analysis [158](#)
Aggregation timeout [80](#)
alias [248](#)
Analyze drop-down [31](#)
Assertion analysis
 View attributes [148](#)
 View list of assertions [146](#)
 View source code [148](#)
Assertions page [145](#)
Attribute Selector [65](#)
Attributes tab [60](#)

B

Bin-Level Merging [274](#)
Block analysis [105](#)
 View attributes [109](#)
 View covered/uncovered blocks [107](#)
 View source code [109](#)
Block page [106](#)
Bubble Diagram [136](#)

C

Check Must Trace [84](#)
CLI commands
 alias [248](#)
 cpu [256](#)
 exclude [188](#)
 exit [256](#)
 help [255](#)
 history [255](#)
 load -refinement [225, 245](#)
 load -run [243](#)
 memory [256](#)
 merge [259](#)
 merge_config [267](#)
 preferences [250](#)
 rank [299](#)
 rank_config [297](#)

report -detail [340](#)
report -list [328](#)
report -summary [331](#)
save -refinement [224, 246](#)
show [244](#)
sourcemap [247](#)
unalias [249](#)
unexclude [223](#)
unload [244](#)
Cover Groups page [154](#)
CoverGroup analysis
 View attributes [162](#)
 View cover bins [157](#)
 View cover items [156](#)
 View list of covergroups [155](#)
 View source [163](#)
cpu [256](#)
Cumulative grade [407, 416](#)

D

Default reviewer [83](#)
Dialog Choices [20](#)

E

exclude [188](#)
Exclude Smart [176](#)
exit [256](#)
Expression analysis [115](#)
 View attributes [121](#)
 View covered/uncovered
 expressions [116](#)
 View expression hierarchy [120](#)
 View SOP table [117](#)
 View source code [119](#)
Expression page [115](#)

F

filter table data [66](#)
FSM analysis
 View arcs [139](#)

View attributes [139](#)
View bubble diagram [136](#)
View covered/uncovered States/
 Transitions [134](#)
View list of FSMs [134](#)
View source code [140](#)
FSM page [133](#)
Full expression in header column [81](#)
Functional coverage [144](#)

H

help [255](#)
history [255](#)
HTML Reports [370](#)

I

IMC Interface [16](#)
IMC window
 Coverage view area [38](#)
 Menu bar [17](#)
 Navigation pages [36](#)
Impacted Entities [178](#)
Impacting Entities [179](#)

J

JasperGold [262](#)

L

Launch IMC
 in batch mode [237, 315](#)
load -refinement [225, 245](#)
load -run [243](#)
Load Runs
 in batch mode [243](#)
 in GUI mode [39](#)
Lookup drop-down [31](#)
Lookup Enclosing Entity in Hierarchy [33](#)
Lookup Entity in Hierarchy [32](#)
Lookup Type [32](#)

M

memory [256](#)
merge [259](#)
Merge Initial model
 empty [288](#)
 primary_run [286](#)
 union_all [287](#)
Merge resolution [272](#)
 initial [272](#)
 union [272](#)
Merge runs
 Merge configuration [267](#)
 Merge metrics data [259](#)
merge_config [267, 272](#)
Merging Coverage Coming from ICC and
 JasperGold (JG) [266](#)
Message Popups [19](#)
Metrics page [43](#)
 Details pane [60](#)
 Information tabs [58](#)
Metrics tab [60](#)
Multi-Dimensional Arrays [203](#)

N

Navigation pages [36](#)

P

parity tree [353](#)
preferences [250, 251](#)

R

rank [299](#)
Rank runs
 Rank configuration [297](#)
 Rank the runs [299](#)
rank_config [297](#)
Refinement
 Exclude Instances or Types [168](#)
 Exclude Specific Items [171](#)
 Exclude Specific Metrics [170](#)
 Exclude Top-Level Instance only [169](#)
 Load Saved Refinements [187](#)
 Save Refinements [186](#)

Un-Exclude Excluded Items [185](#)
Refinement drop-down [35](#)
Refinements
 Load [187](#)
 Save [186](#)
remove columns [64](#)
reorder columns [64](#)
report -detail [340](#)
report -list [328](#)
report -summary [331](#)
report_metrics [391](#)
Reset Settings [22](#)
resize columns [64](#)

S

save -refinement [246](#)
Save Settings [22](#)
search bar [56](#)
Select Attributes [52, 59](#)
Selection History Buttons [36](#)
Show cell suffix [81](#)
show -run [244](#)
Smart Exclusion [177](#)
sort table data [64](#)
Source tab [61](#)
sourcemap [247](#)
Standard merge [272](#)

T

Toggle analysis [124](#)
 View attributes [128](#)
 View source code [128](#)
Toggle page [124](#)
toolbar [26](#)
Tools & Options drop-down [50, 59](#)
Transitions & Arc Sources [134](#)

U

unalias [249](#)
unexclude [223](#)
Union merge [282](#)
unload -run [244](#)